Marketing Insights for E-Commerce Company

Problem statement: A rapidly growing e-commerce company aims to transition from intuition-based marketing to a data-driven approach. By analyzing customer demographics, transaction data, marketing spend, and discount details from 2019, the company seeks to gain a comprehensive understanding of customer behavior. The objectives are to optimize marketing campaigns across various channels, leverage data insights to enhance customer retention, predict customer lifetime value, and ultimately drive sustainable revenue growth.

In [10]:
```
!pip install gdown
```

```
Requirement already satisfied: gdown in c:\users\sinchan\anaconda3\lib\site-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (4.12
.2)
Requirement already satisfied: filelock in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (3.13.1)
Requirement already satisfied: requests[socks] in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (2.3
1.0)
Requirement already satisfied: tqdm in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (4.65.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\sinchan\anaconda3\lib\site-packages (from beautifulsoup
4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\sinchan\anaconda3\lib\site-packages (from re
quests[socks]->gdown) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sinchan\anaconda3\lib\site-packages (from requests[socks
]->gdown) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sinchan\anaconda3\lib\site-packages (from requests
[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sinchan\anaconda3\lib\site-packages (from requests
[socks]->gdown) (2024.2.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in c:\users\sinchan\anaconda3\lib\site-packages (from requ
ests[socks]->gdown) (1.7.1)
Requirement already satisfied: colorama in c:\users\sinchan\anaconda3\lib\site-packages (from tqdm->gdown) (0.4.
6)
```

In [11]:
```
!pip install gdown seaborn lifetimes mlxtend
```

```
Requirement already satisfied: gdown in c:\users\sinchan\anaconda3\lib\site-packages (5.2.0)
Requirement already satisfied: seaborn in c:\users\sinchan\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: lifetimes in c:\users\sinchan\anaconda3\lib\site-packages (0.11.3)
Requirement already satisfied: mlxtend in c:\users\sinchan\anaconda3\lib\site-packages (0.23.1)
Requirement already satisfied: beautifulsoup4 in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (4.12
.2)
Requirement already satisfied: filelock in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (3.13.1)
Requirement already satisfied: requests[socks] in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (2.3
1.0)
Requirement already satisfied: tqdm in c:\users\sinchan\anaconda3\lib\site-packages (from gdown) (4.65.0)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\sinchan\anaconda3\lib\site-packages (from seabor
n) (1.26.4)
Requirement already satisfied: pandas>=0.25 in c:\users\sinchan\anaconda3\lib\site-packages (from seaborn) (2.1.
4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\sinchan\anaconda3\lib\site-packages (from sea
born) (3.8.0)
Requirement already satisfied: scipy>=1.0.0 in c:\users\sinchan\anaconda3\lib\site-packages (from lifetimes) (1.
11.4)
Requirement already satisfied: autograd>=1.2.0 in c:\users\sinchan\anaconda3\lib\site-packages (from lifetimes)
(1.6.2)
Requirement already satisfied: dill>=0.2.6 in c:\users\sinchan\anaconda3\lib\site-packages (from lifetimes) (0.3
.7)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\sinchan\anaconda3\lib\site-packages (from mlxtend
) (1.2.2)
Requirement already satisfied: joblib>=0.13.2 in c:\users\sinchan\anaconda3\lib\site-packages (from mlxtend) (1.
2.0)
Requirement already satisfied: future>=0.15.2 in c:\users\sinchan\anaconda3\lib\site-packages (from autograd>=1.
2.0->lifetimes) (0.18.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
!=3.6.1,>=3.1->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!=3.
6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!=3
.6.1,>=3.1->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sinchan\anaconda3\lib\site-packages (from matplo
tlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas>=0.25->
seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas>=0.25
->seaborn) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\sinchan\anaconda3\lib\site-packages (from scikit
-learn>=1.0.2->mlxtend) (2.2.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\sinchan\anaconda3\lib\site-packages (from beautifulsoup
4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\sinchan\anaconda3\lib\site-packages (from re
quests[socks]->gdown) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sinchan\anaconda3\lib\site-packages (from requests[socks
]->gdown) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sinchan\anaconda3\lib\site-packages (from requests
[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sinchan\anaconda3\lib\site-packages (from requests
[socks]->gdown) (2024.2.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in c:\users\sinchan\anaconda3\lib\site-packages (from requ
ests[socks]->gdown) (1.7.1)
Requirement already satisfied: colorama in c:\users\sinchan\anaconda3\lib\site-packages (from tqdm->gdown) (0.4.
6)
Requirement already satisfied: six>=1.5 in c:\users\sinchan\anaconda3\lib\site-packages (from python-dateutil>=2
.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
```

In [105…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import gdown
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```python
from scipy.stats import ttest_1samp, ttest_ind,ttest_rel
from scipy.stats import chisquare,chi2,chi2_contingency
from scipy.stats import f_oneway
from scipy.stats import levene,kruskal
from statsmodels.graphics.gofplots import qqplot
```

In [5]:

```python
# Download the dataset from Google Drive
url = 'https://drive.google.com/drive/folders/1VXaZSDFqN_Zi3FxucRlthz97Etl1CYfJ?usp=sharing'
```

```
gdown.download_folder(url, quiet=False)
```

```
Retrieving folder contents
Processing file 1nKb67meAvoFNRG_VueNXh_-k_2F9Oisc Customers.csv
Processing file 1fI7Kg4iXh3GTxFlto4n0UQ-5nOzAM78- Dataset Description.docx
Processing file 144wrGLTgzCm8FZdl0rasbjsm6dKG_yvB Discount_Coupon.csv
Processing file 1xgzPmbSCU8KM6Lt1rJXLFADHxgsy9u4t Marketing_Spend.csv
Processing file 1F8EPu3t_GbXX3BQ3OQSXsoieeSsCR5ly Online_Sales.csv
Processing file 1CmZ0j83nKoNeOqbGgZHSa92-kdXHkXIP Tax_amount.csv
Retrieving folder contents completed
Building directory structure
Building directory structure completed
Downloading...
From: https://drive.google.com/uc?id=1nKb67meAvoFNRG_VueNXh_-k_2F9Oisc
To: C:\Users\sinchan\Block 2 project\Customers.csv
100%|██████████████████████████████████████████████████████████████████████████████| 31.8k/31.8k [00:00<00:00, 2.
03MB/s]
Downloading...
From: https://drive.google.com/uc?id=1fI7Kg4iXh3GTxFlto4n0UQ-5nOzAM78-
To: C:\Users\sinchan\Block 2 project\Dataset Description.docx
100%|███████████████████████████████████████████████████████████████████████████████| 7.52k/7.52k [00:00<00:00, 9
53kB/s]
Downloading...
From: https://drive.google.com/uc?id=144wrGLTgzCm8FZdl0rasbjsm6dKG_yvB
To: C:\Users\sinchan\Block 2 project\Discount_Coupon.csv
100%|███████████████████████████████████████████████████████████████████████████████| 4.92k/4.92k [00:00<00:00, 7
81kB/s]
Downloading...
From: https://drive.google.com/uc?id=1xgzPmbSCU8KM6Lt1rJXLFADHxgsy9u4t
To: C:\Users\sinchan\Block 2 project\Marketing_Spend.csv
100%|██████████████████████████████████████████████████████████████████████████████| 8.67k/8.67k [00:00<00:00, 3.
32MB/s]
Downloading...
From: https://drive.google.com/uc?id=1F8EPu3t_GbXX3BQ3OQSXsoieeSsCR5ly
To: C:\Users\sinchan\Block 2 project\Online_Sales.csv
100%|██████████████████████████████████████████████████████████████████████████████| 5.24M/5.24M [00:00<00:00, 5.
29MB/s]
Downloading...
From: https://drive.google.com/uc?id=1CmZ0j83nKoNeOqbGgZHSa92-kdXHkXIP
To: C:\Users\sinchan\Block 2 project\Tax_amount.csv
100%|███████████████████████████████████████████████████████████████████████████████████████████| 297/297 [00:00<?
, ?B/s]
Download completed
```

Out[5]:
```
['C:\\Users\\sinchan\\Block 2 project\\Customers.csv',
 'C:\\Users\\sinchan\\Block 2 project\\Dataset Description.docx',
 'C:\\Users\\sinchan\\Block 2 project\\Discount_Coupon.csv',
 'C:\\Users\\sinchan\\Block 2 project\\Marketing_Spend.csv',
 'C:\\Users\\sinchan\\Block 2 project\\Online_Sales.csv',
 'C:\\Users\\sinchan\\Block 2 project\\Tax_amount.csv']
```

In [7]:
```
# Load the datasets into pandas dataframes
customer_df = pd.read_csv(r'C:\\Users\\sinchan\\Block 2 project\\Customers.csv')
taxamount_df = pd.read_csv(r'C:\Users\sinchan\Block 2 project\Tax_amount.csv')
marketing_df = pd.read_csv(r'C:\Users\sinchan\Block 2 project\Marketing_Spend.csv')
discount_df = pd.read_csv(r'C:\Users\sinchan\Block 2 project\Discount_Coupon.csv')
onlinesales_df = pd.read_csv(r'C:\\Users\\sinchan\\Block 2 project\\Online_Sales.csv')
```

In [41]:
```
customer_df.head()
```

Out[41]:

	CustomerID	Gender	Location	Tenure_Months
0	17850	M	Chicago	12
1	13047	M	California	43
2	12583	M	Chicago	33
3	13748	F	California	30
4	15100	M	California	49

In [46]:
```
customer_df.nunique()
```

Out[46]:
```
CustomerID       1468
Gender              2
Location            5
Tenure_Months      49
dtype: int64
```

In [42]:
```
taxamount_df.head()
```

Out[42]:

	Product_Category	GST
0	Nest-USA	10%
1	Office	10%
2	Apparel	18%
3	Bags	18%
4	Drinkware	18%

In [49]:
```
taxamount_df.nunique()
```

Out[49]:
```
Product_Category    20
GST                  4
dtype: int64
```

In [43]:
```
marketing_df.head()
```

Out[43]:

	Date	Offline_Spend	Online_Spend
0	1/1/2019	4500	2424.50
1	1/2/2019	4500	3480.36
2	1/3/2019	4500	1576.38
3	1/4/2019	4500	2928.55
4	1/5/2019	4500	4055.30

In [50]:
```
marketing_df.nunique()
```

Out[50]:
```
Date             365
Offline_Spend     11
Online_Spend     365
dtype: int64
```

In [44]:
```
discount_df.head()
```

Out[44]:

	Month	Product_Category	Coupon_Code	Discount_pct
0	Jan	Apparel	SALE10	10
1	Feb	Apparel	SALE20	20
2	Mar	Apparel	SALE30	30
3	Jan	Nest-USA	ELEC10	10
4	Feb	Nest-USA	ELEC20	20

In [51]:
```
discount_df.nunique()
```

Out[51]:
```
Month               12
Product_Category    17
Coupon_Code         48
Discount_pct         3
dtype: int64
```

In [45]:
```
onlinesales_df.head()
```

Out[45]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price
0	17850	16679	1/1/2019	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71
1	17850	16680	1/1/2019	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71
2	17850	16681	1/1/2019	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05
3	17850	16682	1/1/2019	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53
4	17850	16682	1/1/2019	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50

In [47]:
```
onlinesales_df.nunique()
```

Out[47]:
```
CustomerID              1468
Transaction_ID         25061
Transaction_Date         365
Product_SKU             1145
Product_Description      404
Product_Category          20
Quantity                 151
Avg_Price                546
Delivery_Charges         267
Coupon_Status              3
dtype: int64
```

EXPLORATORY DATA ANALYSIS

CHECKING SHAPE

In [17]:
```
print(customer_df.shape)
print(taxamount_df.shape)
print(marketing_df.shape)
print(discount_df.shape)
print(onlinesales_df.shape)
```

```
(1468, 4)
(20, 2)
(365, 3)
(204, 4)
(52924, 10)
```

In [18]:
```
print(customer_df.info(),"\n\n")
print(taxamount_df.info(),"\n\n")
print(marketing_df.info(),"\n\n")
print(discount_df.info(),"\n\n")
print(onlinesales_df.info(),"\n\n")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1468 entries, 0 to 1467
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   CustomerID     1468 non-null   int64 
 1   Gender         1468 non-null   object
 2   Location       1468 non-null   object
 3   Tenure_Months  1468 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 46.0+ KB
None


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Product_Category  20 non-null     object
 1   GST               20 non-null     object
dtypes: object(2)
memory usage: 452.0+ bytes
None


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           365 non-null    object 
 1   Offline_Spend  365 non-null    int64  
 2   Online_Spend   365 non-null    float64
dtypes: float64(1), int64(1), object(1)
memory usage: 8.7+ KB
None


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 204 entries, 0 to 203
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Month             204 non-null    object
 1   Product_Category  204 non-null    object
 2   Coupon_Code       204 non-null    object
 3   Discount_pct      204 non-null    int64 
dtypes: int64(1), object(3)
memory usage: 6.5+ KB
None


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52924 entries, 0 to 52923
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   CustomerID           52924 non-null  int64  
 1   Transaction_ID       52924 non-null  int64  
 2   Transaction_Date     52924 non-null  object 
 3   Product_SKU          52924 non-null  object 
 4   Product_Description  52924 non-null  object 
 5   Product_Category     52924 non-null  object 
 6   Quantity             52924 non-null  int64  
 7   Avg_Price            52924 non-null  float64
 8   Delivery_Charges     52924 non-null  float64
 9   Coupon_Status        52924 non-null  object 
dtypes: float64(2), int64(3), object(5)
memory usage: 4.0+ MB
None
```

In [26]:
```
customer_df.describe(include="all")
```

Out[26]:

	CustomerID	Gender	Location	Tenure_Months
count	1468.000000	1468	1468	1468.000000
unique	NaN	2	5	NaN
top	NaN	F	California	NaN
freq	NaN	934	464	NaN
mean	15314.386240	NaN	NaN	25.912125
std	1744.000367	NaN	NaN	13.959667
min	12346.000000	NaN	NaN	2.000000
25%	13830.500000	NaN	NaN	14.000000
50%	15300.000000	NaN	NaN	26.000000
75%	16882.250000	NaN	NaN	38.000000
max	18283.000000	NaN	NaN	50.000000

In [22]:
```
taxamount_df.describe(include="all")
```

Out[22]:

	Product_Category	GST
count	20	20
unique	20	4
top	Nest-USA	10%
freq	1	7

In [23]:
```
marketing_df.describe(include="all")
```

Out[23]:

	Date	Offline_Spend	Online_Spend
count	365	365.000000	365.000000
unique	365	NaN	NaN
top	1/1/2019	NaN	NaN
freq	1	NaN	NaN
mean	NaN	2843.561644	1905.880740
std	NaN	952.292448	808.856853
min	NaN	500.000000	320.250000
25%	NaN	2500.000000	1258.600000
50%	NaN	3000.000000	1881.940000
75%	NaN	3500.000000	2435.120000
max	NaN	5000.000000	4556.930000

In [24]:
```
discount_df.describe(include="all")
```

Out[24]:

	Month	Product_Category	Coupon_Code	Discount_pct
count	204	204	204	204.000000
unique	12	17	48	NaN
top	Jan	Apparel	EXTRA10	NaN
freq	17	12	8	NaN
mean	NaN	NaN	NaN	20.000000
std	NaN	NaN	NaN	8.185052
min	NaN	NaN	NaN	10.000000
25%	NaN	NaN	NaN	10.000000
50%	NaN	NaN	NaN	20.000000
75%	NaN	NaN	NaN	30.000000
max	NaN	NaN	NaN	30.000000

In [25]:
```
onlinesales_df.describe(include="all")
```

Out[25]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity
count	52924.00000	52924.000000	52924	52924	52924	52924	52924.000000
unique	NaN	NaN	365	1145	404	20	NaN
top	NaN	NaN	11/27/2019	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Apparel	NaN
freq	NaN	NaN	335	3511	3511	18126	NaN
mean	15346.70981	32409.825675	NaN	NaN	NaN	NaN	4.497638
std	1766.55602	8648.668977	NaN	NaN	NaN	NaN	20.104711
min	12346.00000	16679.000000	NaN	NaN	NaN	NaN	1.000000
25%	13869.00000	25384.000000	NaN	NaN	NaN	NaN	1.000000
50%	15311.00000	32625.500000	NaN	NaN	NaN	NaN	1.000000
75%	16996.25000	39126.250000	NaN	NaN	NaN	NaN	2.000000
max	18283.00000	48497.000000	NaN	NaN	NaN	NaN	900.000000

In [27]:
```
customer_df.isnull().sum()
```
Out[27]:
```
CustomerID       0
Gender           0
Location         0
Tenure_Months    0
dtype: int64
```
In [29]:
```
marketing_df.isnull().sum()
```
Out[29]:
```
Date             0
Offline_Spend    0
Online_Spend     0
dtype: int64
```
In [30]:
```
taxamount_df.isnull().sum()
```
Out[30]:
```
Product_Category    0
GST                 0
dtype: int64
```
In [31]:
```
discount_df.isnull().sum()
```
Out[31]:
```
Month               0
Product_Category    0
Coupon_Code         0
Discount_pct        0
dtype: int64
```
In [32]:
```
onlinesales_df.isnull().sum()
```
Out[32]:
```
CustomerID             0
Transaction_ID         0
Transaction_Date       0
Product_SKU            0
Product_Description    0
Product_Category       0
Quantity               0
Avg_Price              0
Delivery_Charges       0
Coupon_Status          0
dtype: int64
```
In [28]:
```
customer_df.isna().sum()
```
Out[28]:
```
CustomerID       0
Gender           0
Location         0
Tenure_Months    0
dtype: int64
```
In [33]:
```
marketing_df.isna().sum()
```
Out[33]:
```
Date             0
Offline_Spend    0
Online_Spend     0
dtype: int64
```
In [34]:
```
taxamount_df.isna().sum()
```

```
Out[34]: Product_Category    0
         GST                 0
         dtype: int64
```

```
In [35]: discount_df.isna().sum()
```

```
Out[35]: Month               0
         Product_Category    0
         Coupon_Code         0
         Discount_pct        0
         dtype: int64
```

```
In [36]: onlinesales_df.isna().sum()
```

```
Out[36]: CustomerID             0
         Transaction_ID         0
         Transaction_Date       0
         Product_SKU            0
         Product_Description    0
         Product_Category       0
         Quantity               0
         Avg_Price              0
         Delivery_Charges       0
         Coupon_Status          0
         dtype: int64
```

No null or missing values found

Non Grpahical Analysis

```
In [40]: customer_df.duplicated().sum()
```

```
Out[40]: 0
```

```
In [52]: marketing_df.duplicated().sum()
```

```
Out[52]: 0
```

```
In [53]: taxamount_df.duplicated().sum()
```

```
Out[53]: 0
```

```
In [54]: discount_df.duplicated().sum()
```

```
Out[54]: 0
```

```
In [55]: onlinesales_df.duplicated().sum()
```

```
Out[55]: 0
```

```
In [12]: # Define a function to perform value counts on categorical columns
def value_counts_for_categorical(df):
    for column in df.columns:
        if df[column].dtype == 'object' or df[column].dtype.name == 'category':
            print(f"\nValue counts for {column} in dataframe:")
            print(df[column].value_counts())
            print("-" * 50)

# Apply the function to each dataframe
print("Customer Dataframe")
value_counts_for_categorical(customer_df)

print("Tax Amount Dataframe")
value_counts_for_categorical(taxamount_df)

print("Marketing Dataframe")
value_counts_for_categorical(marketing_df)

print("Discount Dataframe")
value_counts_for_categorical(discount_df)

print("Online Dataframe")
value_counts_for_categorical(onlinesales_df)
```

```
Customer Dataframe

Value counts for Gender in dataframe:
Gender
```

```
F    934
M    534
Name: count, dtype: int64
--------------------------------------------------

Value counts for Location in dataframe:
Location
California       464
Chicago          456
New York         324
New Jersey       149
Washington DC     75
Name: count, dtype: int64
--------------------------------------------------
Tax Amount Dataframe

Value counts for Product_Category in dataframe:
Product_Category
Nest-USA                1
Office                  1
Accessories             1
Android                 1
Housewares              1
More Bags               1
Gift Cards              1
Bottles                 1
Google                  1
Backpacks               1
Nest-Canada             1
Fun                     1
Waze                    1
Headgear                1
Notebooks & Journals    1
Lifestyle               1
Drinkware               1
Bags                    1
Apparel                 1
Nest                    1
Name: count, dtype: int64
--------------------------------------------------
Marketing Dataframe
Discount Dataframe

Value counts for Month in dataframe:
Month
Jan    17
Feb    17
Mar    17
Apr    17
May    17
Jun    17
Jul    17
Aug    17
Sep    17
Oct    17
Nov    17
Dec    17
Name: count, dtype: int64
--------------------------------------------------

Value counts for Product_Category in dataframe:
Product_Category
Apparel                 12
Waze                    12
Notebooks & Journals    12
Gift Cards              12
Accessories             12
Housewares              12
Nest-Canada             12
Bottles                 12
Nest                    12
Nest-USA                12
Headgear                12
Notebooks               12
Bags                    12
Lifestyle               12
Drinkware               12
Office                  12
Android                 12
Name: count, dtype: int64
--------------------------------------------------

Value counts for Coupon_Code in dataframe:
```

```
Coupon_Code
EXTRA10    8
EXTRA20    8
EXTRA30    8
SALE10     4
ACC20      4
BT20       4
BT30       4
NCA10      4
NCA20      4
NCA30      4
HOU10      4
HOU20      4
HOU30      4
ACC10      4
GC10       4
ACC30      4
WEMP30     4
GC20       4
GC30       4
NJ10       4
NJ20       4
NJ30       4
AND10      4
AND20      4
BT10       4
WEMP10     4
WEMP20     4
SALE20     4
SALE30     4
ELEC10     4
ELEC20     4
ELEC30     4
OFF10      4
OFF20      4
OFF30      4
AIO10      4
AIO20      4
AIO30      4
NOTES10    4
NOTES20    4
NOTES30    4
HGEAR10    4
HGEAR20    4
HGEAR30    4
NE10       4
NE20       4
NE30       4
AND30      4
Name: count, dtype: int64
--------------------------------------------------
Online Dataframe

Value counts for Product_SKU in dataframe:
Product_SKU
GGOENEBJ079499    3511
GGOENEBQ078999    3328
GGOENEBB078899    3230
GGOENEBQ079099    1361
GGOENEBQ084699    1089
                  ...
GGOEAAWQ063049       1
GGOEYAEB030014       1
GGOEGAHB057413       1
GGOEWALJ083416       1
GGOEGOCJ093999       1
Name: count, Length: 1145, dtype: int64
--------------------------------------------------

Value counts for Product_Description in dataframe:
Product_Description
Nest Learning Thermostat 3rd Gen-USA - Stainless Steel         3511
Nest Cam Outdoor Security Camera - USA                         3328
Nest Cam Indoor Security Camera - USA                          3230
Google Sunglasses                                              1523
Nest Protect Smoke + CO White Battery Alarm-USA                1361
                                                               ...
Google Tee Red                                                    2
Google Women's Colorblock Tee White                               1
Compact Journal with Recycled Pages                               1
Android Women's Short Sleeve Tri-blend Badge Tee Light Blue       1
Google Large Standard Journal Grey                                1
Name: count, Length: 404, dtype: int64
```

```
-------------------------------------------------

Value counts for Product_Category in dataframe:
Product_Category
Apparel                 18126
Nest-USA                14013
Office                   6513
Drinkware                3483
Lifestyle                3092
Nest                     2198
Bags                     1882
Headgear                  771
Notebooks & Journals      749
Waze                      554
Nest-Canada               317
Bottles                   268
Accessories               234
Fun                       160
Gift Cards                159
Housewares                122
Google                    105
Backpacks                  89
More Bags                  46
Android                    43
Name: count, dtype: int64
-------------------------------------------------

Value counts for Coupon_Status in dataframe:
Coupon_Status
Clicked     26926
Used        17904
Not Used     8094
Name: count, dtype: int64
-------------------------------------------------
```

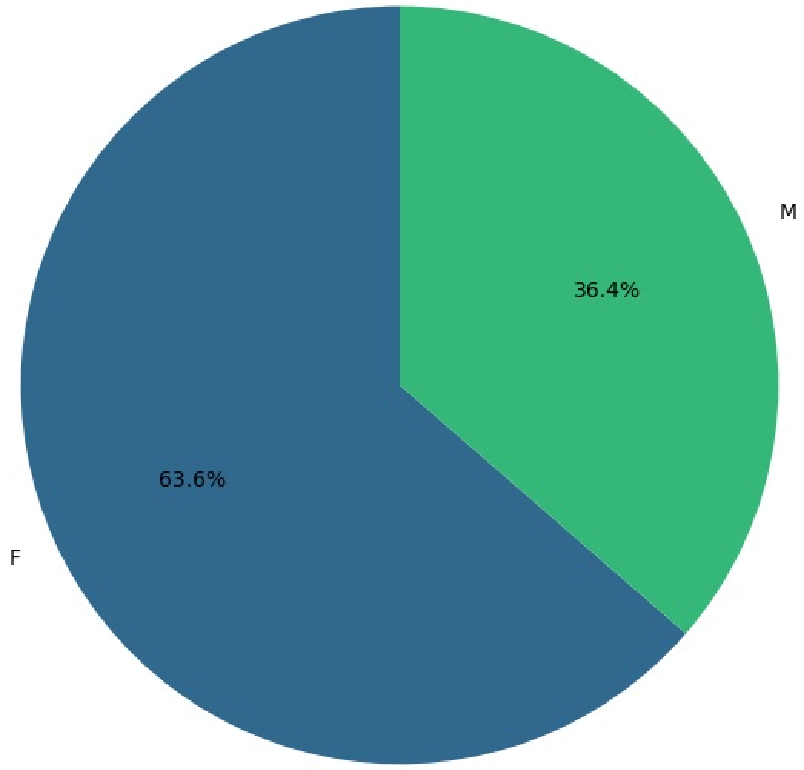
RELATIONSHIP BETWEEN VARIABLES

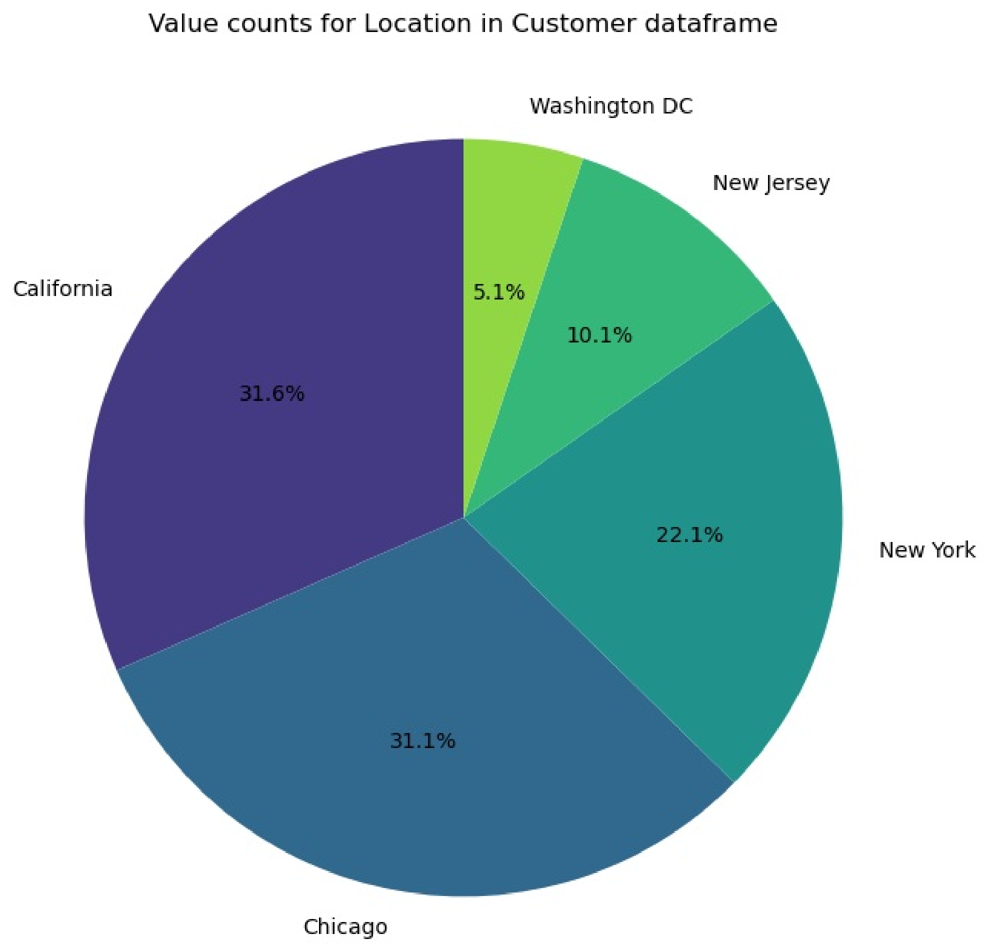
In [15]:

```
def plot_value_counts(df, df_name):
    for column in df.columns:
        if df[column].dtype == 'object' or df[column].dtype.name == 'category':
            value_counts = df[column].value_counts()
            if value_counts.size <= 10:  # If there are 10 or fewer unique categories, plot a pie chart
                plt.figure(figsize=(8, 8))
                value_counts.plot.pie(autopct='%1.1f%%', startangle=90, colors=sns.color_palette('viridis', valu
                plt.title(f'Value counts for {column} in {df_name} dataframe')
                plt.ylabel('')  # Hide y-label for pie chart
                plt.show()
            else:  # Otherwise, plot a bar chart
                plt.figure(figsize=(10, 6))
                sns.countplot(data=df, y=column, order=value_counts.index, palette='viridis')
                plt.title(f'Value counts for {column} in {df_name} dataframe')
                plt.xlabel('Count')
                plt.ylabel(column)
                plt.show()

# Apply the function to each dataframe
print("Customer Dataframe")
plot_value_counts(customer_df, "Customer")

print("Tax Amount Dataframe")
plot_value_counts(taxamount_df, "Tax Amount")

print("Marketing Dataframe")
plot_value_counts(marketing_df, "Marketing")

print("Discount Dataframe")
plot_value_counts(discount_df, "Discount")

print("Online Dataframe")
plot_value_counts(onlinesales_df, "Online")
```

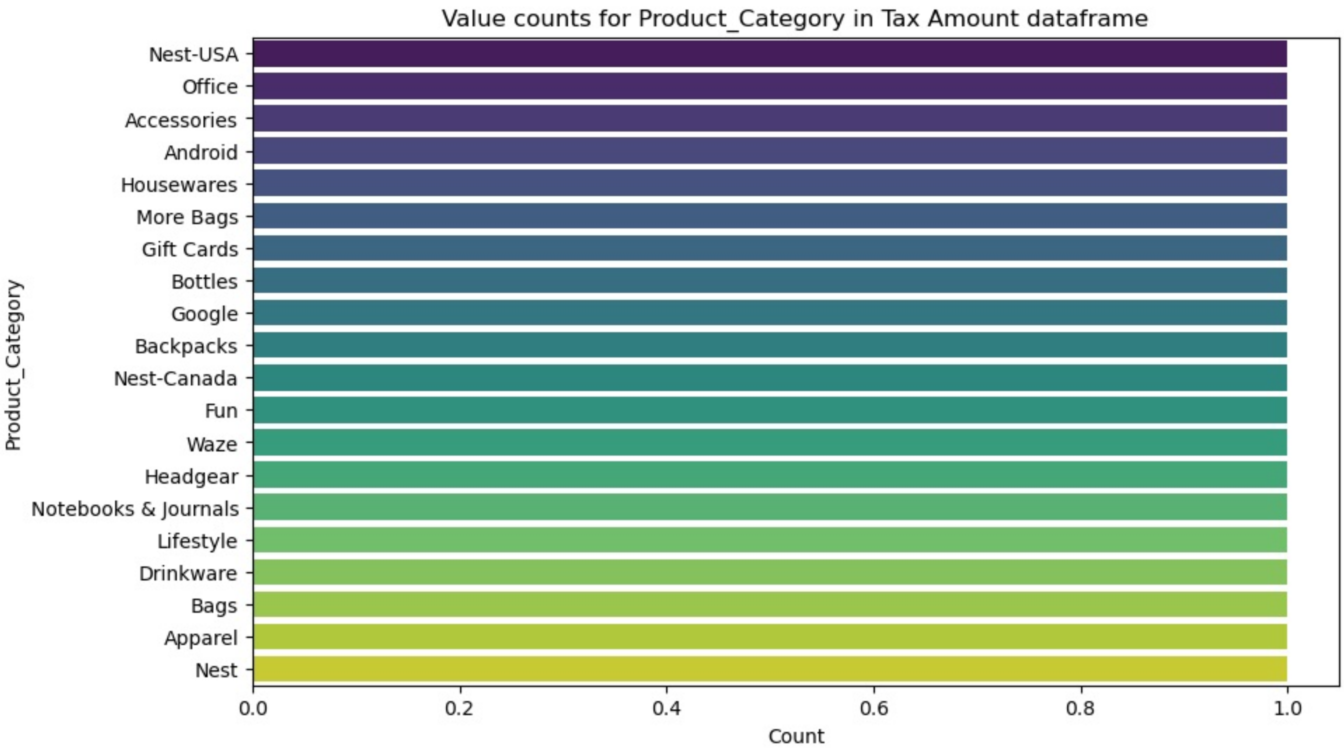
```
Customer Dataframe
```

Value counts for Gender in Customer dataframe

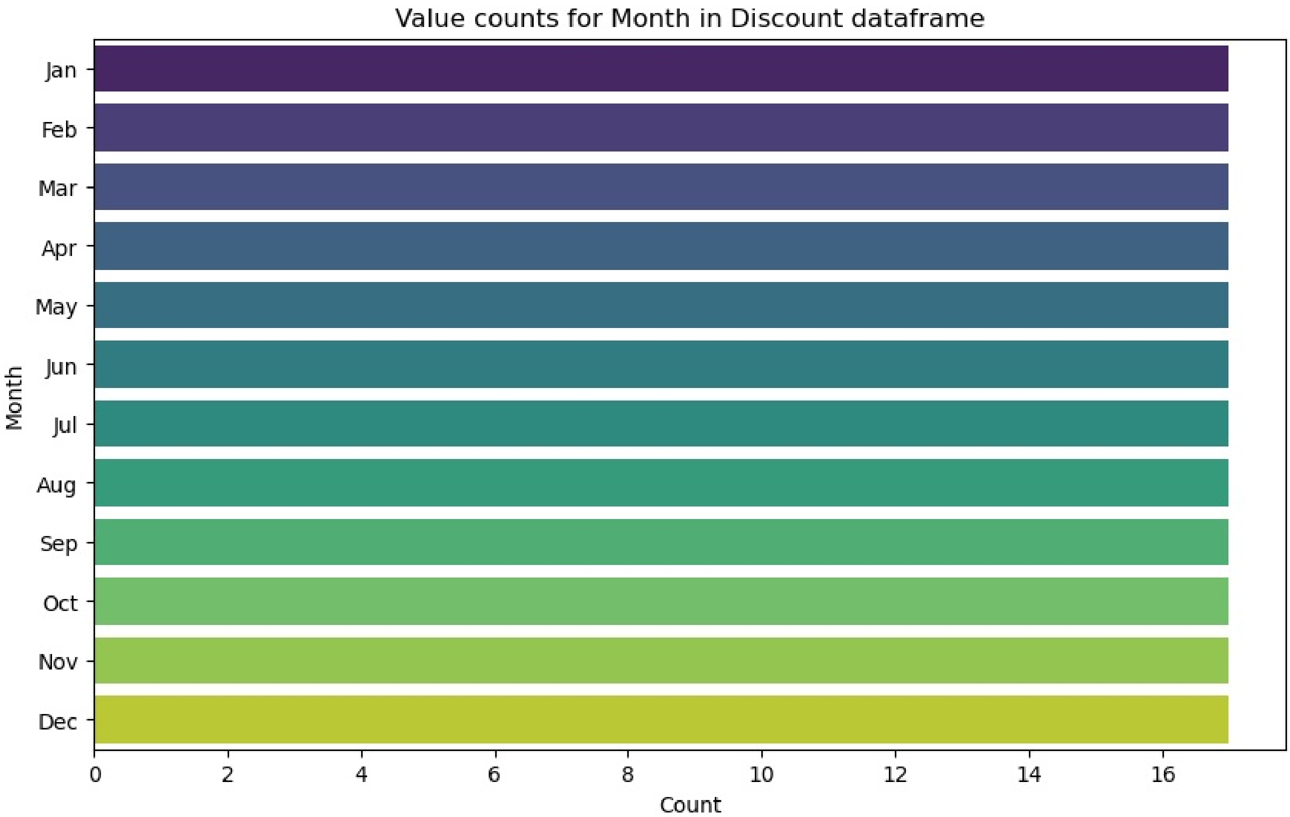


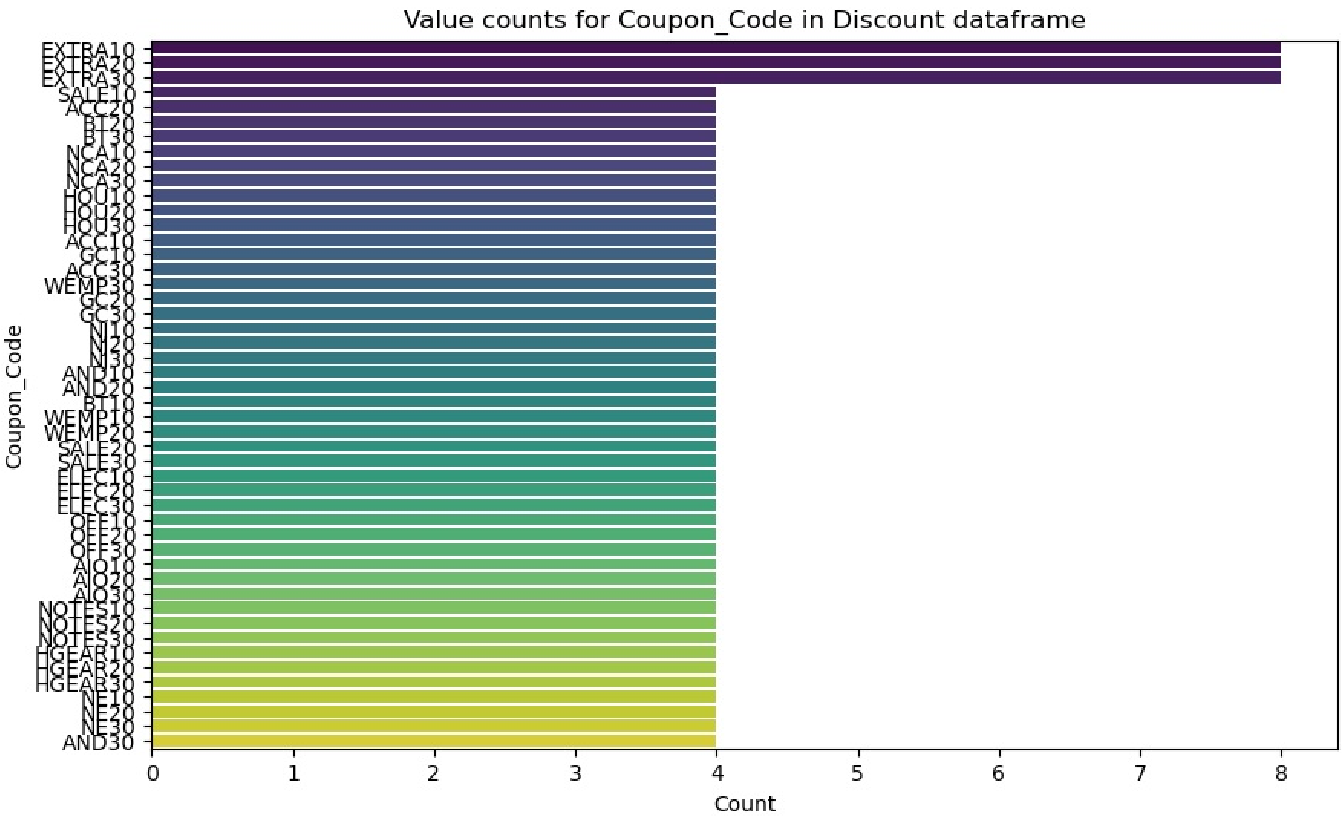



```
Tax Amount Dataframe
```

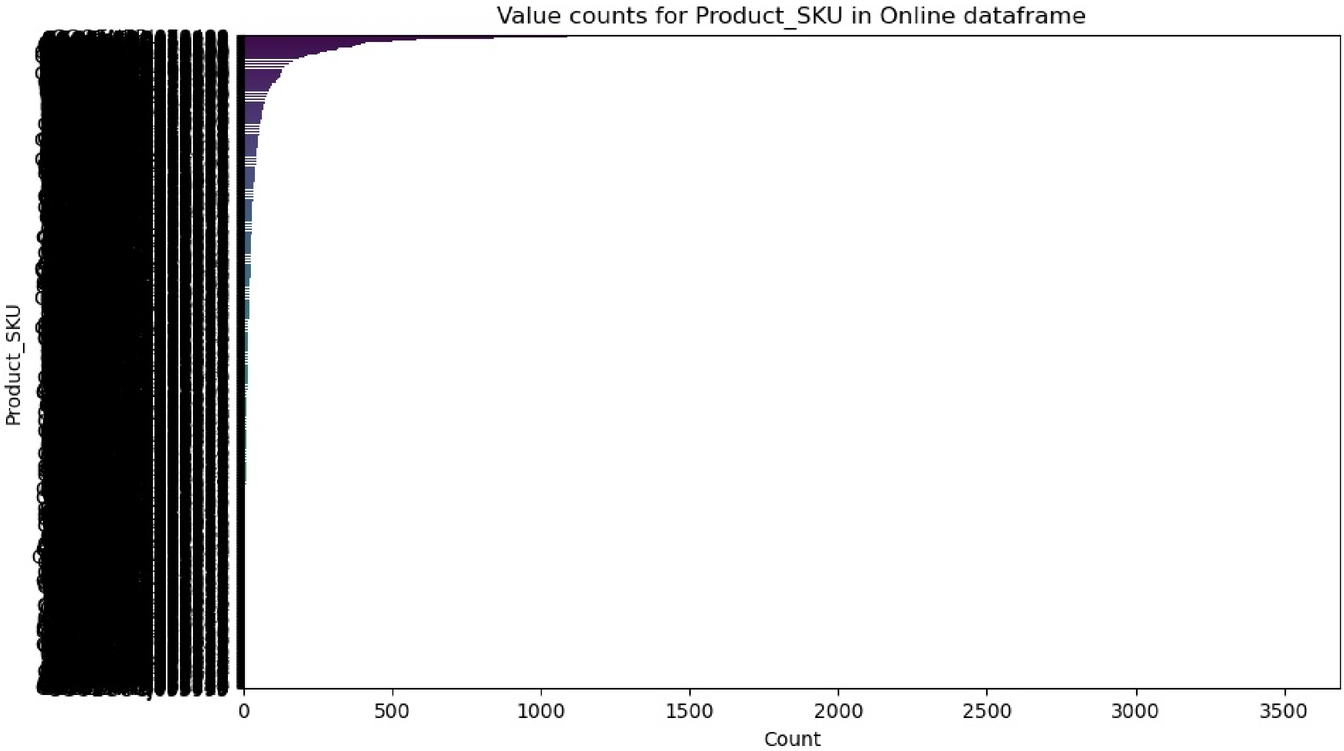


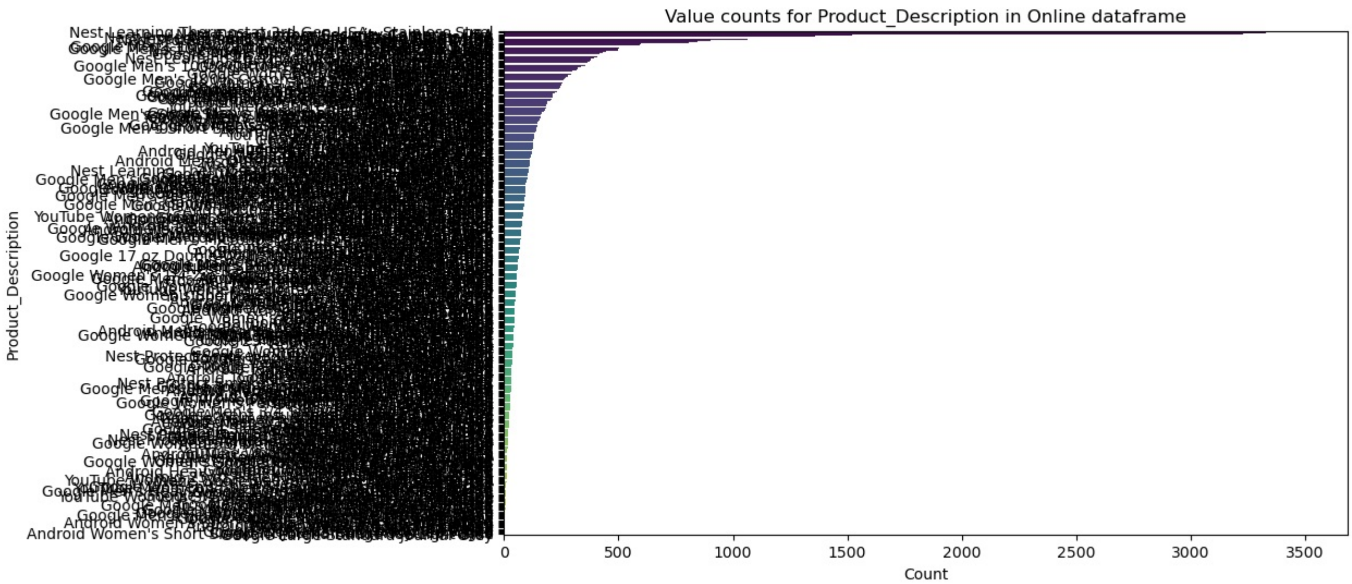

```
Marketing Dataframe
Discount Dataframe
```

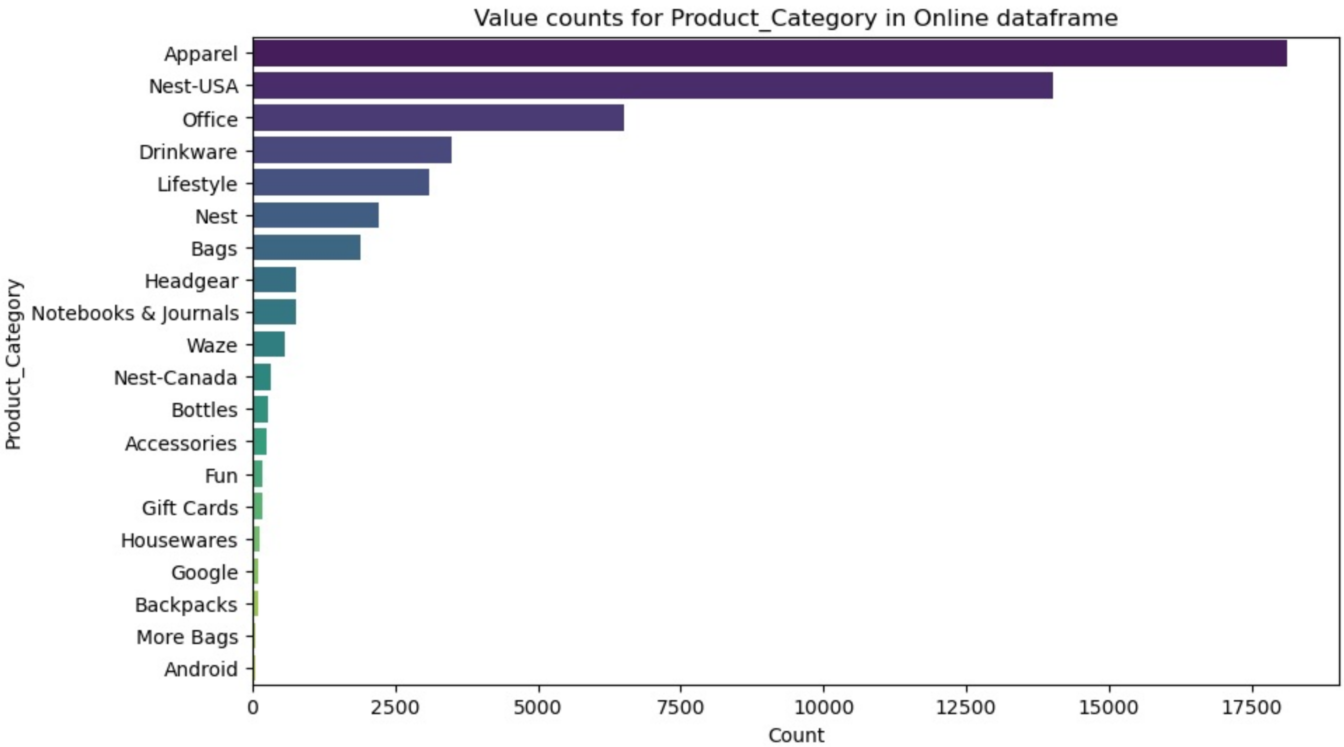




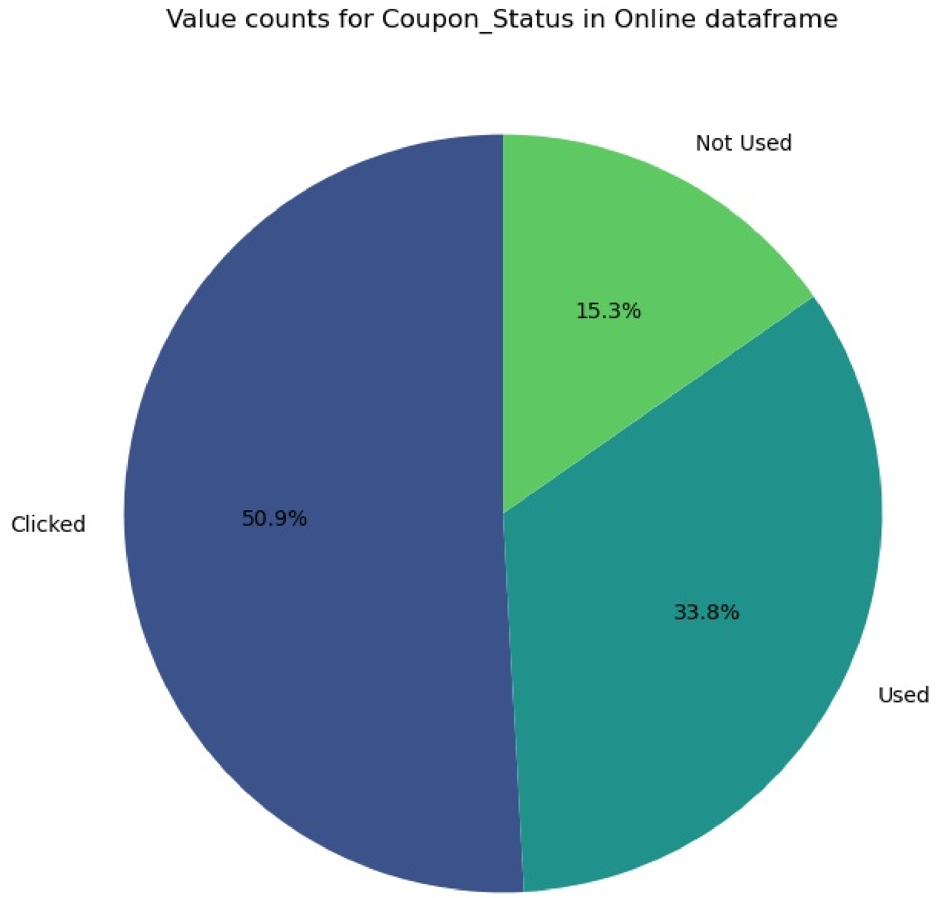

```
Online Dataframe
```


Value counts for Product_SKU in Online dataframe
Product_SKU
0
500
1000
1500
2000
2500
3000
3500
Count


Value counts for Product_Description in Online dataframe
Product_Description
0
500
1000
1500
2000
2500
3000
3500
Count



Value counts for Product_Category in Online dataframe
Apparel
Nest-USA
Office
Drinkware
Lifestyle
Nest
Bags
Headgear
Notebooks & Journals
Waze
Nest-Canada
Bottles
Accessories
Fun
Gift Cards
Housewares
Google
Backpacks
More Bags
Android
Product_Category
0
2500
5000
7500
10000
12500
15000
17500
Count



In [ ]:
```
discount_df contains the discount information and taxamount_df contains the GST information, merge them into the
```

It is crucial to begin by calculating the invoice amount or revenue for each transaction using this formula, This establishes the foundation for revenue analysis. For that we will perform some necessary calculations.

In [110…
```
# Convert GST and Discount_pct to float for calculation
#taxamount_df['GST'] = taxamount_df['GST'].str.rstrip('%').astype('float') / 100.0
#discount_df['Discount_pct'] = discount_df['Discount_pct'].astype('float') / 100.0
```

In [14]:
```python
# Merge onlinesales_df with discount_df to get Discount_pct
sales_discount_df = pd.merge(onlinesales_df, discount_df, on='Product_Category', how='left')

# Merge the result with taxamount_df to get GST
sales_full_df = pd.merge(sales_discount_df, taxamount_df, on='Product_Category', how='left')

# Calculate Invoice Value
sales_full_df['Invoice_Value'] = (
    sales_full_df['Quantity'] * sales_full_df['Avg_Price'] *
    (1 - sales_full_df['Discount_pct']) *
    (1 + sales_full_df['GST']) +
    sales_full_df['Delivery_Charges']
)

# Display the first few rows of the result
sales_full_df[['CustomerID', 'Transaction_ID', 'Product_Category', 'Quantity', 'Avg_Price', 'Discount_pct', 'GS
```

Out[14]:

	CustomerID	Transaction_ID	Product_Category	Quantity	Avg_Price	Discount_pct	GST	Delivery_Charges	Invoice_Value
0	17850	16679	Nest-USA	1	153.71	0.1	0.1	6.5	158.6729
1	17850	16679	Nest-USA	1	153.71	0.2	0.1	6.5	141.7648
2	17850	16679	Nest-USA	1	153.71	0.3	0.1	6.5	124.8567
3	17850	16679	Nest-USA	1	153.71	0.1	0.1	6.5	158.6729
4	17850	16679	Nest-USA	1	153.71	0.2	0.1	6.5	141.7648
5	17850	16679	Nest-USA	1	153.71	0.3	0.1	6.5	124.8567
6	17850	16679	Nest-USA	1	153.71	0.1	0.1	6.5	158.6729
7	17850	16679	Nest-USA	1	153.71	0.2	0.1	6.5	141.7648
8	17850	16679	Nest-USA	1	153.71	0.3	0.1	6.5	124.8567
9	17850	16679	Nest-USA	1	153.71	0.1	0.1	6.5	158.6729

In [ ]:
```python
Now we need to remove extra white space from both male and female values of Gender column in customer data set
```

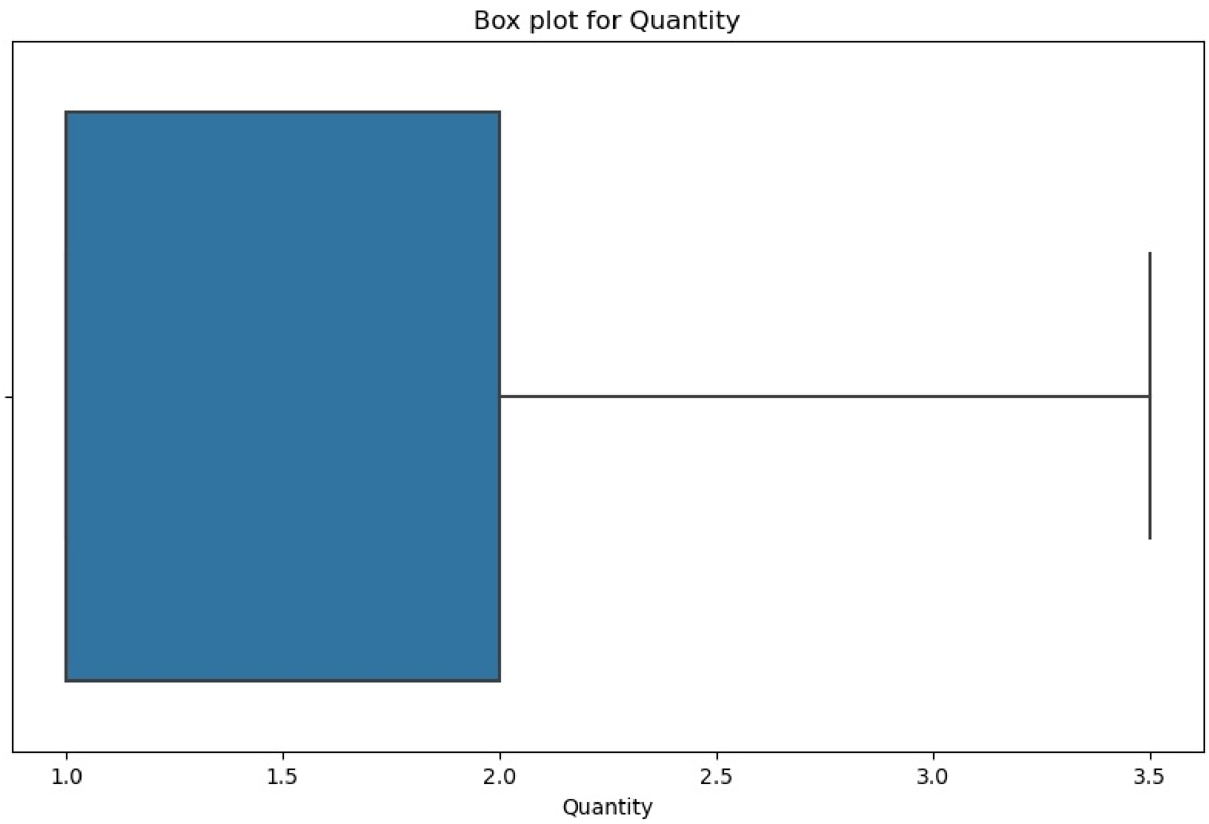
In [16]:
```python
customer_df['Gender'] = customer_df['Gender'].replace({'Male ': 'Male', 'Female ': 'Female'})
```

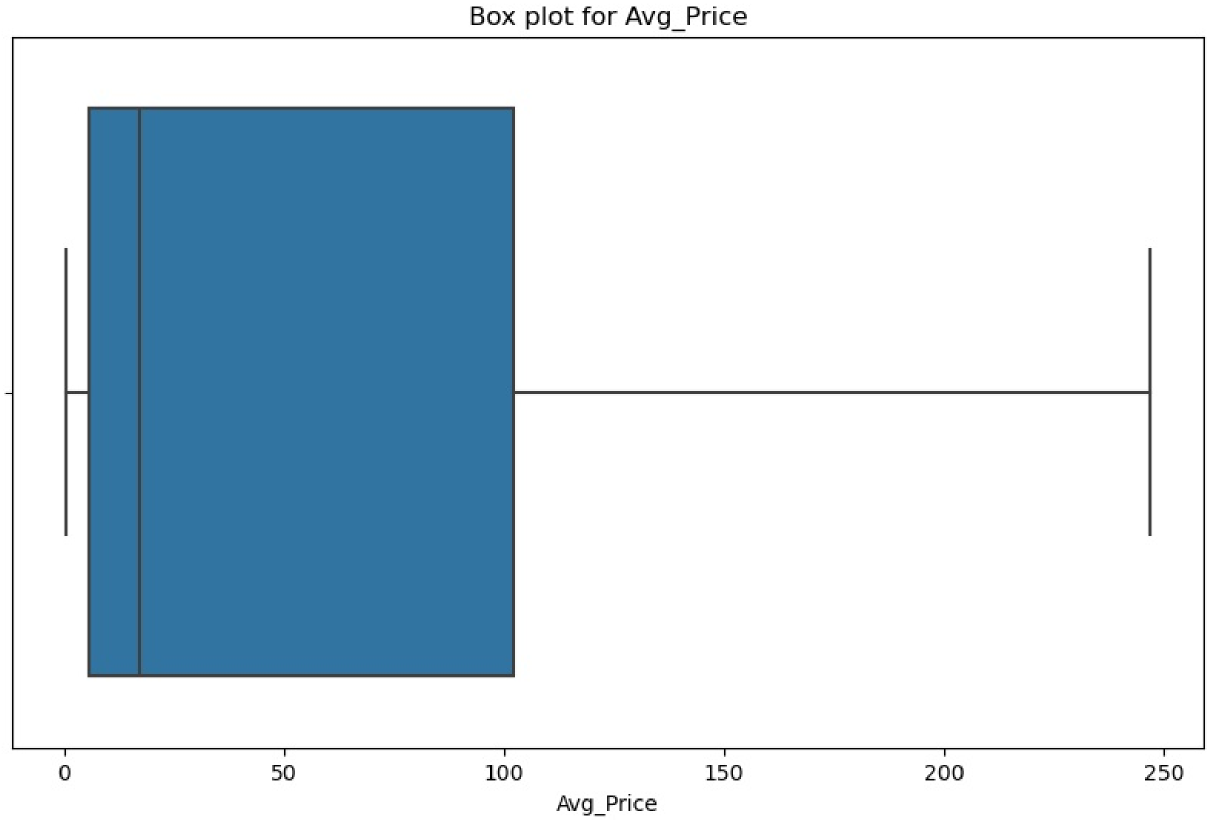
In [ ]:
```python
5. Identifying and Handling Outliers
```

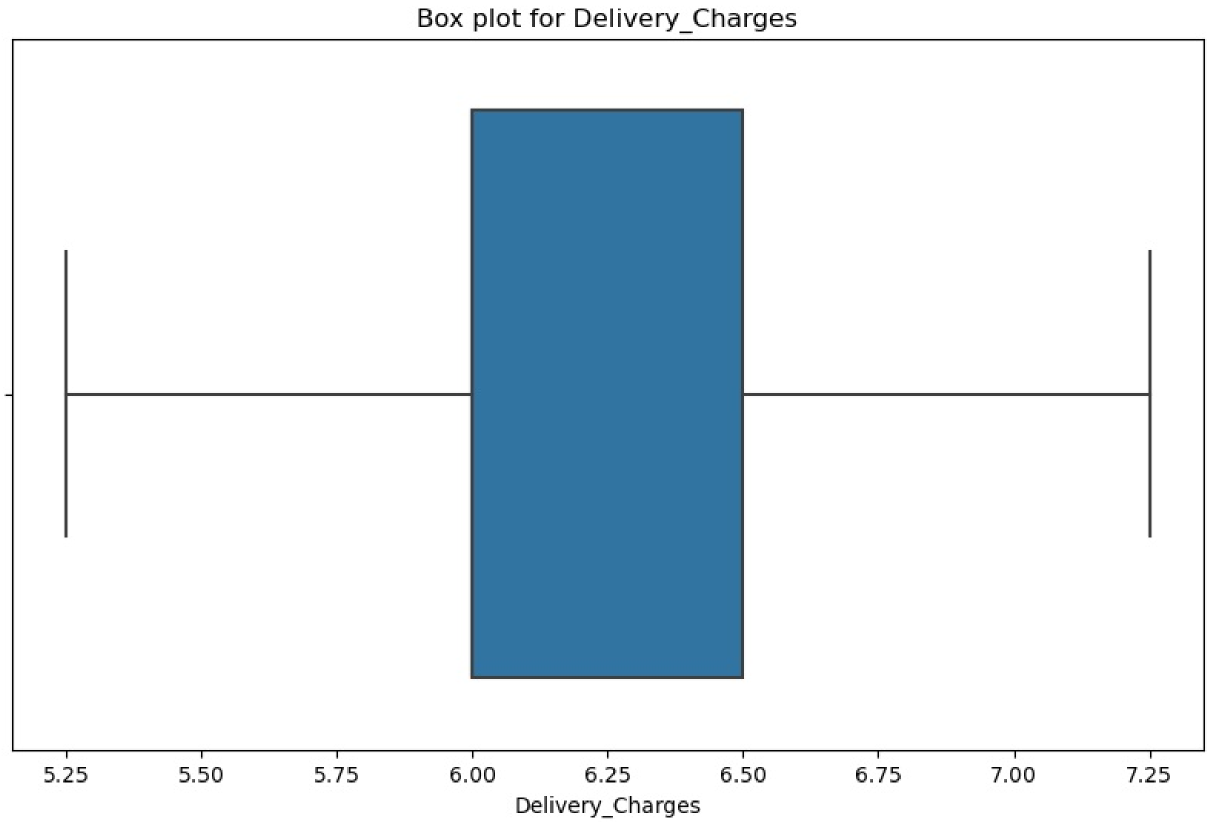
In [21]:
```python
def detect_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

# Detecting outliers in numerical columns of onlinesales_df
numerical_columns_onlinesales = ['Quantity', 'Avg_Price', 'Delivery_Charges']
for col in numerical_columns_onlinesales:
    outliers = detect_outliers(onlinesales_df, col)
    print(f'Outliers in {col}:\n', outliers, "\n\n")

# Detecting outliers in numerical columns of marketing_df
numerical_columns_marketing = ['Offline_Spend', 'Online_Spend']
for col in numerical_columns_marketing:
    outliers = detect_outliers(marketing_df, col)
    print(f'Outliers in {col}:\n', outliers, "\n\n")

# Handling outliers (example: capping the outliers)
for col in numerical_columns_onlinesales:
    Q1 = onlinesales_df[col].quantile(0.25)
    Q3 = onlinesales_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    onlinesales_df[col] = np.where(onlinesales_df[col] < lower_bound, lower_bound, onlinesales_df[col])
    onlinesales_df[col] = np.where(onlinesales_df[col] > upper_bound, upper_bound, onlinesales_df[col])

for col in numerical_columns_marketing:
    Q1 = marketing_df[col].quantile(0.25)
    Q3 = marketing_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    marketing_df[col] = np.where(marketing_df[col] < lower_bound, lower_bound, marketing_df[col])
    marketing_df[col] = np.where(marketing_df[col] > upper_bound, upper_bound, marketing_df[col])
```

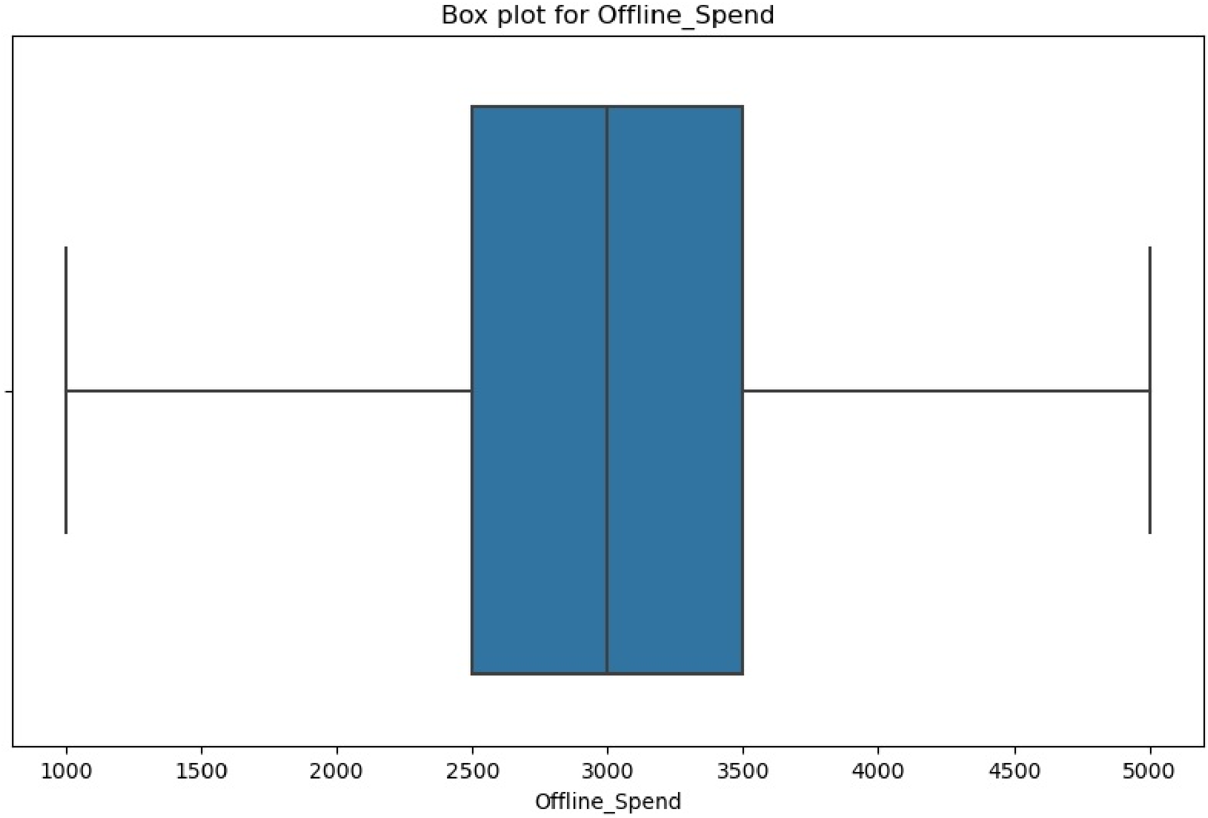
```python
# Visualize to check for outliers again
for col in numerical_columns_onlinesales:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=onlinesales_df[col])
    plt.title(f'Box plot for {col}')
    plt.show()

for col in numerical_columns_marketing:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=marketing_df[col])
    plt.title(f'Box plot for {col}')
    plt.show()
```

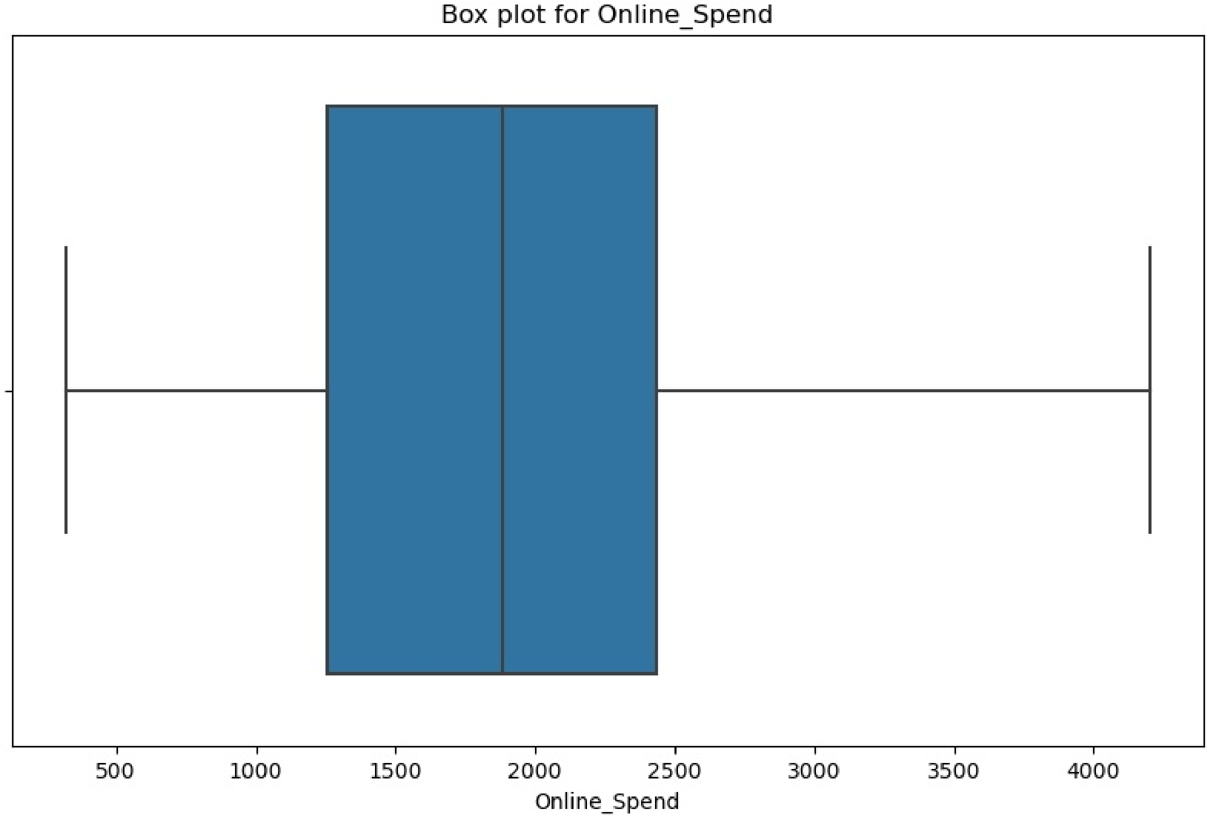
```
Outliers in Quantity:
 Empty DataFrame
Columns: [CustomerID, Transaction_ID, Transaction_Date, Product_SKU, Product_Description, Product_Category, Quan
tity, Avg_Price, Delivery_Charges, Coupon_Status]
Index: []


Outliers in Avg_Price:
 Empty DataFrame
Columns: [CustomerID, Transaction_ID, Transaction_Date, Product_SKU, Product_Description, Product_Category, Quan
tity, Avg_Price, Delivery_Charges, Coupon_Status]
Index: []


Outliers in Delivery_Charges:
 Empty DataFrame
Columns: [CustomerID, Transaction_ID, Transaction_Date, Product_SKU, Product_Description, Product_Category, Quan
tity, Avg_Price, Delivery_Charges, Coupon_Status]
Index: []


Outliers in Offline_Spend:
 Empty DataFrame
Columns: [Date, Offline_Spend, Online_Spend]
Index: []


Outliers in Online_Spend:
 Empty DataFrame
Columns: [Date, Offline_Spend, Online_Spend]
Index: []
```




Box plot for Avg_Price
0
50
100
150
200
250
Avg_Price


Box plot for Delivery_Charges
5.25
5.50
5.75
6.00
6.25
6.50
6.75
7.00
7.25
Delivery_Charges





In [ ]:
```
univariate n bivariate
```

Exploratory Data Analysis (EDA):

Customer Acquisition & Retention: Analyze trends in customer acquisition and churn across different customer demographics (gender, location, tenure)

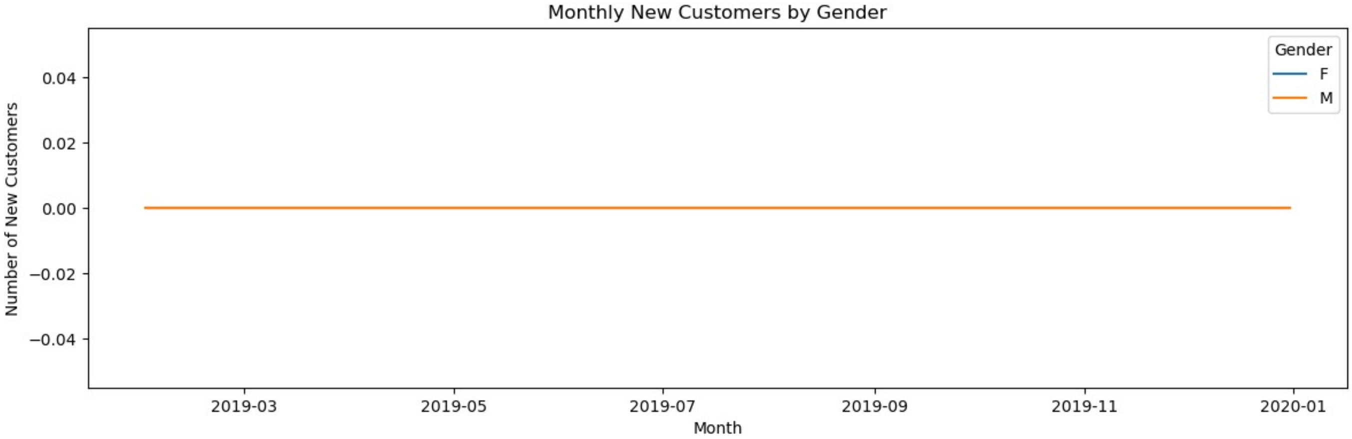
and timeframes (monthly).

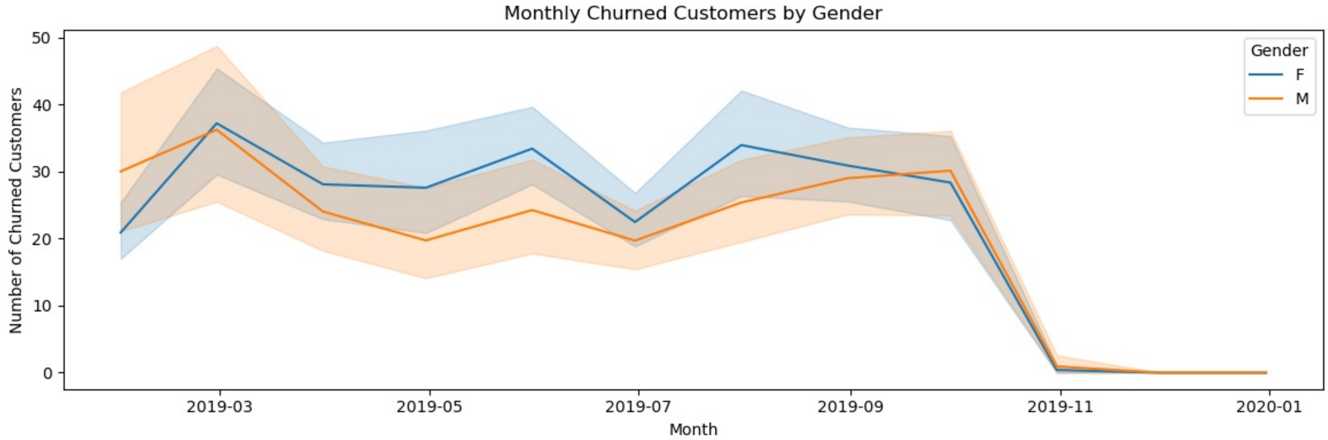
Tools like time series analysis and segmentation can be helpful here.

In [111…

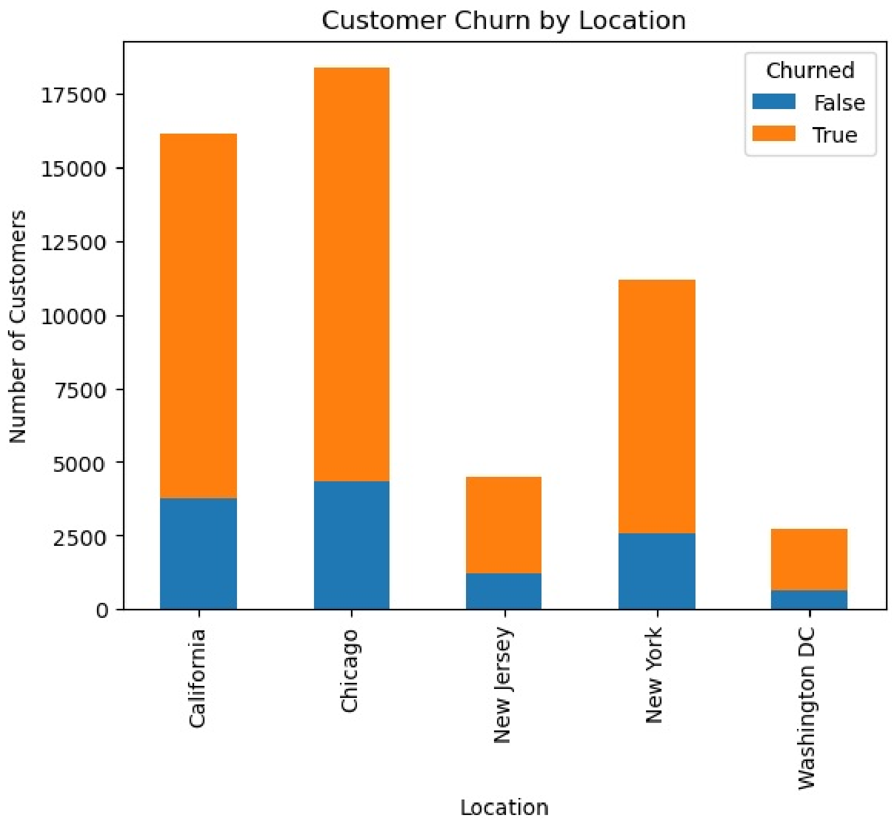
```
# Ensure 'Transaction_Date' is datetime format in onlinesales_df
onlinesales_df['Transaction_Date'] = pd.to_datetime(onlinesales_df['Transaction_Date'])

# Merge onlinesales_df with customer_df to get demographic information
merged_df = pd.merge(onlinesales_df, customer_df, on='CustomerID')

# Identify new customers (first appearance in the dataset)
merged_df['First_Transaction_Date'] = merged_df.groupby('CustomerID')['Transaction_Date'].transform('min')

# Identify churned customers (no transactions in the last 3 months)
last_transaction_date = merged_df['Transaction_Date'].max()
churn_period = pd.Timedelta(days=90)
merged_df['Churned'] = merged_df['Transaction_Date'] < (last_transaction_date - churn_period)

# Create a summary DataFrame for monthly analysis
monthly_summary = merged_df.groupby([pd.Grouper(key='Transaction_Date', freq='M'), 'Gender', 'Location', 'Tenure
    new_customers=('First_Transaction_Date', lambda x: x.eq(x.name).sum()),
    churned_customers=('Churned', 'sum')
).reset_index()

# Handle inf values by replacing them with NaN
monthly_summary.replace([np.inf, -np.inf], np.nan, inplace=True)

# Plot trends in customer acquisition and churn
plt.figure(figsize=(12, 8))

# Monthly new customers
plt.subplot(2, 1, 1)
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='new_customers', hue='Gender')
plt.title('Monthly New Customers by Gender')
plt.xlabel('Month')
plt.ylabel('Number of New Customers')

# Monthly churned customers
plt.subplot(2, 1, 2)
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='churned_customers', hue='Gender')
plt.title('Monthly Churned Customers by Gender')
plt.xlabel('Month')
plt.ylabel('Number of Churned Customers')

plt.tight_layout()
plt.show()

# Additional analysis: churn by location, tenure, and gender
# Group by location, tenure, and gender
location_summary = merged_df.groupby(['Location', 'Churned']).size().unstack(fill_value=0)
tenure_summary = merged_df.groupby(['Tenure_Months', 'Churned']).size().unstack(fill_value=0)
gender_summary = merged_df.groupby(['Gender', 'Churned']).size().unstack(fill_value=0)

# Plot churn by location
plt.figure(figsize=(10, 6))
location_summary.plot(kind='bar', stacked=True)
plt.title('Customer Churn by Location')
plt.xlabel('Location')
plt.ylabel('Number of Customers')
plt.show()

# Plot churn by tenure
plt.figure(figsize=(10, 6))
tenure_summary.plot(kind='bar', stacked=True)
plt.title('Customer Churn by Tenure')
plt.xlabel('Tenure (Months)')
plt.ylabel('Number of Customers')
plt.show()

# Plot churn by gender
plt.figure(figsize=(10, 6))
```

```
gender_summary.plot(kind='bar', stacked=True)
plt.title('Customer Churn by Gender')
plt.xlabel('Gender')
plt.ylabel('Number of Customers')
plt.show()
```

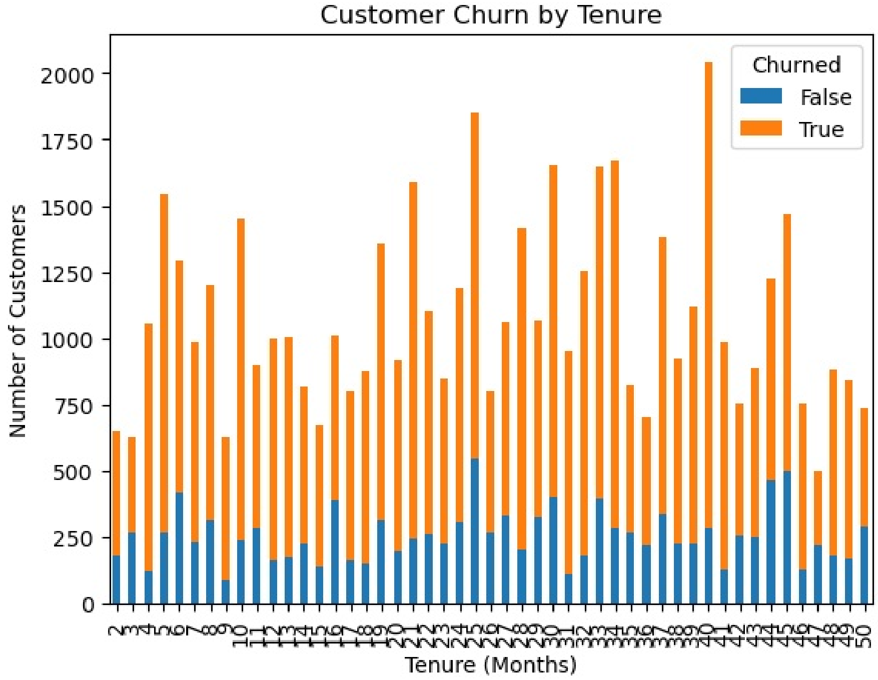




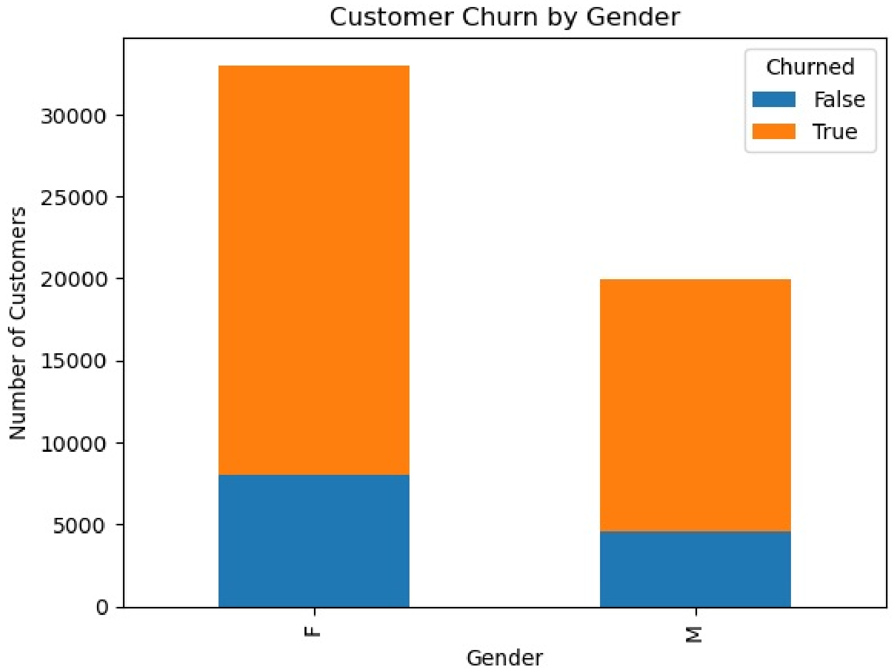
<Figure size 1000x600 with 0 Axes>



<Figure size 1000x600 with 0 Axes>




```
<Figure size 1000x600 with 0 Axes>
```



Explore the relationship between marketing spend (online & offline) and customer behavior (orders, revenue) to assess campaign effectiveness. Utilize techniques like hypothesis testing to validate your findings.

In [32]:

```
# Merge onlinesales_df with discount_df and taxamount_df to get Discount_pct and GST
merged_df = pd.merge(onlinesales_df, discount_df, on='Product_Category', how='left')
merged_df = pd.merge(merged_df, taxamount_df, on='Product_Category', how='left')

# Calculate Invoice Value
merged_df['Invoice_Value'] = ((merged_df['Quantity'] * merged_df['Avg_Price']) *
                              (1 - merged_df['Discount_pct']) *
                              (1 + merged_df['GST'])) + merged_df['Delivery_Charges']
```

```python
# Merge with customer_df to get customer information
merged_df = pd.merge(merged_df, customer_df, on='CustomerID', how='left')

# Summarize monthly revenue and orders
monthly_summary = merged_df.groupby(pd.Grouper(key='Transaction_Date', freq='M')).agg(
    revenue=('Invoice_Value', 'sum'),
    orders=('Transaction_ID', 'count')
).reset_index()

# Merge monthly summary with marketing_df
monthly_summary = pd.merge(monthly_summary, marketing_df, left_on='Transaction_Date', right_on='Month', how='le
print("Monthly Summary columns:", monthly_summary.columns)
```

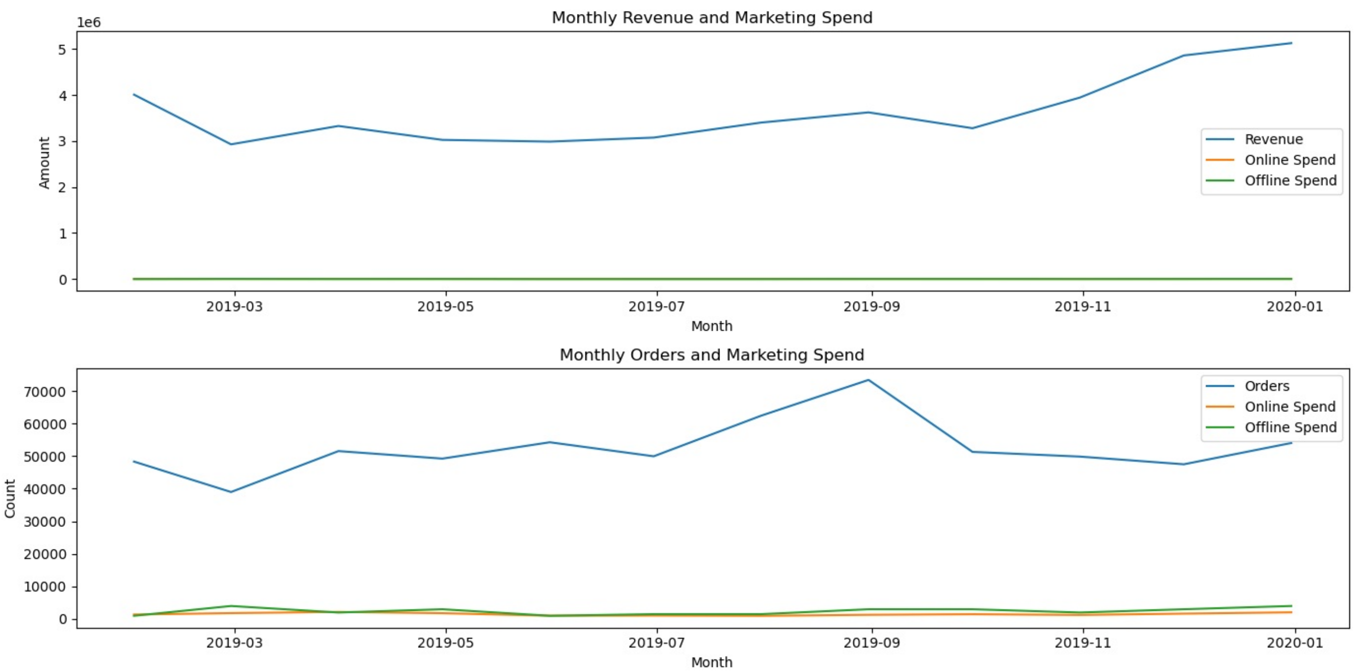
```
Monthly Summary columns: Index(['Transaction_Date', 'revenue', 'orders', 'Month', 'Offline_Spend',
       'Online_Spend'],
      dtype='object')
```

In [ ]:

```
Step 2: Analysis of Marketing Spend and Customer Behavior
```

In [106…

```python
# Plot monthly revenue and marketing spend
plt.figure(figsize=(14, 7))

plt.subplot(2, 1, 1)
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='revenue', label='Revenue')
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='Online_Spend', label='Online Spend')
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='Offline_Spend', label='Offline Spend')
plt.title('Monthly Revenue and Marketing Spend')
plt.xlabel('Month')
plt.ylabel('Amount')
plt.legend()

plt.subplot(2, 1, 2)
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='orders', label='Orders')
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='Online_Spend', label='Online Spend')
sns.lineplot(data=monthly_summary, x='Transaction_Date', y='Offline_Spend', label='Offline Spend')
plt.title('Monthly Orders and Marketing Spend')
plt.xlabel('Month')
plt.ylabel('Count')
plt.legend()

plt.tight_layout()
plt.show()
```



In [ ]:

```
Step 3: Hypothesis Testing
```

In [34]:

```python
# Hypothesis Testing: Correlation between marketing spend and revenue/orders

# Calculate correlation coefficients
from scipy import stats
correlation_revenue_online = monthly_summary[['revenue', 'Online_Spend']].corr().iloc[0, 1]
correlation_revenue_offline = monthly_summary[['revenue', 'Offline_Spend']].corr().iloc[0, 1]
correlation_orders_online = monthly_summary[['orders', 'Online_Spend']].corr().iloc[0, 1]
correlation_orders_offline = monthly_summary[['orders', 'Offline_Spend']].corr().iloc[0, 1]

print(f"Correlation between Revenue and Online Spend: {correlation_revenue_online}")
print(f"Correlation between Revenue and Offline Spend: {correlation_revenue_offline}")
print(f"Correlation between Orders and Online Spend: {correlation_orders_online}")
```

```python
print(f"Correlation between Orders and Offline Spend: {correlation_orders_offline}")

# Perform hypothesis testing
revenue_online_corr, revenue_online_p = stats.pearsonr(monthly_summary['revenue'], monthly_summary['Online_Spend
revenue_offline_corr, revenue_offline_p = stats.pearsonr(monthly_summary['revenue'], monthly_summary['Offline_Sp
orders_online_corr, orders_online_p = stats.pearsonr(monthly_summary['orders'], monthly_summary['Online_Spend']
orders_offline_corr, orders_offline_p = stats.pearsonr(monthly_summary['orders'], monthly_summary['Offline_Spend

print(f"\nRevenue and Online Spend: Correlation={revenue_online_corr}, P-value={revenue_online_p}")
print(f"Revenue and Offline Spend: Correlation={revenue_offline_corr}, P-value={revenue_offline_p}")
print(f"Orders and Online Spend: Correlation={orders_online_corr}, P-value={orders_online_p}")
print(f"Orders and Offline Spend: Correlation={orders_offline_corr}, P-value={orders_offline_p}")
```

```
Correlation between Revenue and Online Spend: 0.3035094158799748
Correlation between Revenue and Offline Spend: 0.29385029077909114
Correlation between Orders and Online Spend: -0.3659409811520193
Correlation between Orders and Offline Spend: -0.14251920700905918

Revenue and Online Spend: Correlation=0.3035094158799747, P-value=0.3375435775461572
Revenue and Offline Spend: Correlation=0.293850290779091, P-value=0.3539006634217138
Orders and Online Spend: Correlation=-0.36594098115201934, P-value=0.24206143273713215
Orders and Offline Spend: Correlation=-0.14251920700905887, P-value=0.6585957703384702
```

In [ ]:

```python
#HYPOTHESIS3 TESTING
```

In [35]:

```python
# Interpret the results
alpha = 0.05  # significance level

if revenue_online_p < alpha:
    print("Reject the null hypothesis: There is a significant correlation between online marketing spend and re
else:
    print("Fail to reject the null hypothesis: There is no significant correlation between online marketing spe

if revenue_offline_p < alpha:
    print("Reject the null hypothesis: There is a significant correlation between offline marketing spend and r
else:
    print("Fail to reject the null hypothesis: There is no significant correlation between offline marketing sp

if orders_online_p < alpha:
    print("Reject the null hypothesis: There is a significant correlation between online marketing spend and or
else:
    print("Fail to reject the null hypothesis: There is no significant correlation between online marketing spe

if orders_offline_p < alpha:
    print("Reject the null hypothesis: There is a significant correlation between offline marketing spend and o
else:
    print("Fail to reject the null hypothesis: There is no significant correlation between offline marketing sp
```

```
Fail to reject the null hypothesis: There is no significant correlation between online marketing spend and reven
ue.
Fail to reject the null hypothesis: There is no significant correlation between offline marketing spend and reve
nue.
Fail to reject the null hypothesis: There is no significant correlation between online marketing spend and order
s.
Fail to reject the null hypothesis: There is no significant correlation between offline marketing spend and orde
rs.
```

To analyze how discounts and promotions affect revenue and customer engagement, we can investigate KPIs like average order value (AOV), customer acquisition cost (CAC), and their relationship with different discount structures. Here's how we can approach this analysis:

Steps to Analyze Discount Effects: Calculate Average Order Value (AOV):

AOV is typically calculated as total revenue divided by the number of orders. It gives insight into the average amount customers are spending per order. Analyze Revenue Impact of Discounts:

Compare total revenue with and without discounts applied to understand the direct impact of discounts on sales. Evaluate Customer Acquisition Cost (CAC):

CAC is the cost incurred to acquire a new customer. It can be calculated by dividing the total costs associated with acquisition efforts by the number of new customers acquired. Perform Discount Sensitivity Analysis:

We will analyze how different levels of discounts (e.g., percentage discounts, fixed discounts, conditional discounts) affect AOV, CAC, and overall revenue.

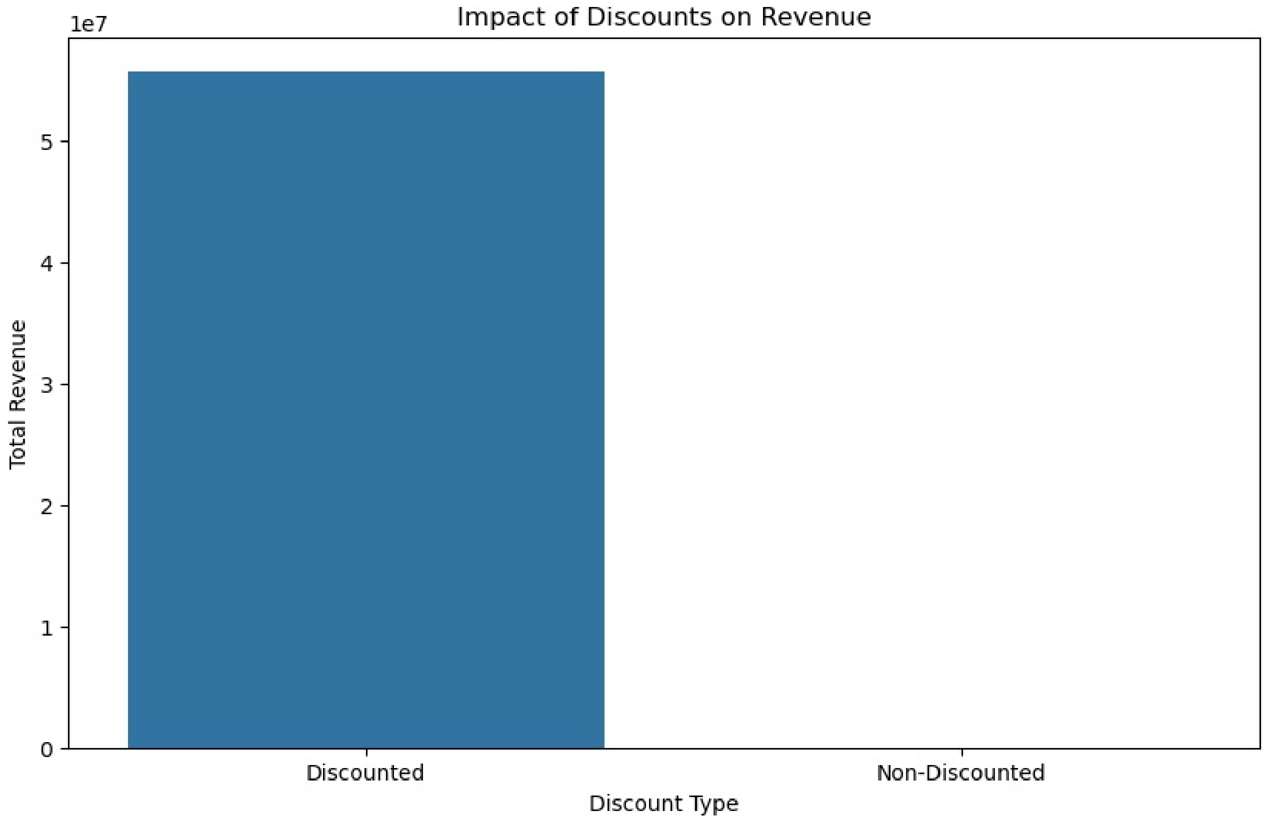
Then we will Conduct hypothesis tests (e.g., t-tests) to determine if there are statistically significant differences in AOV, CAC, or revenue between different discount levels or types.

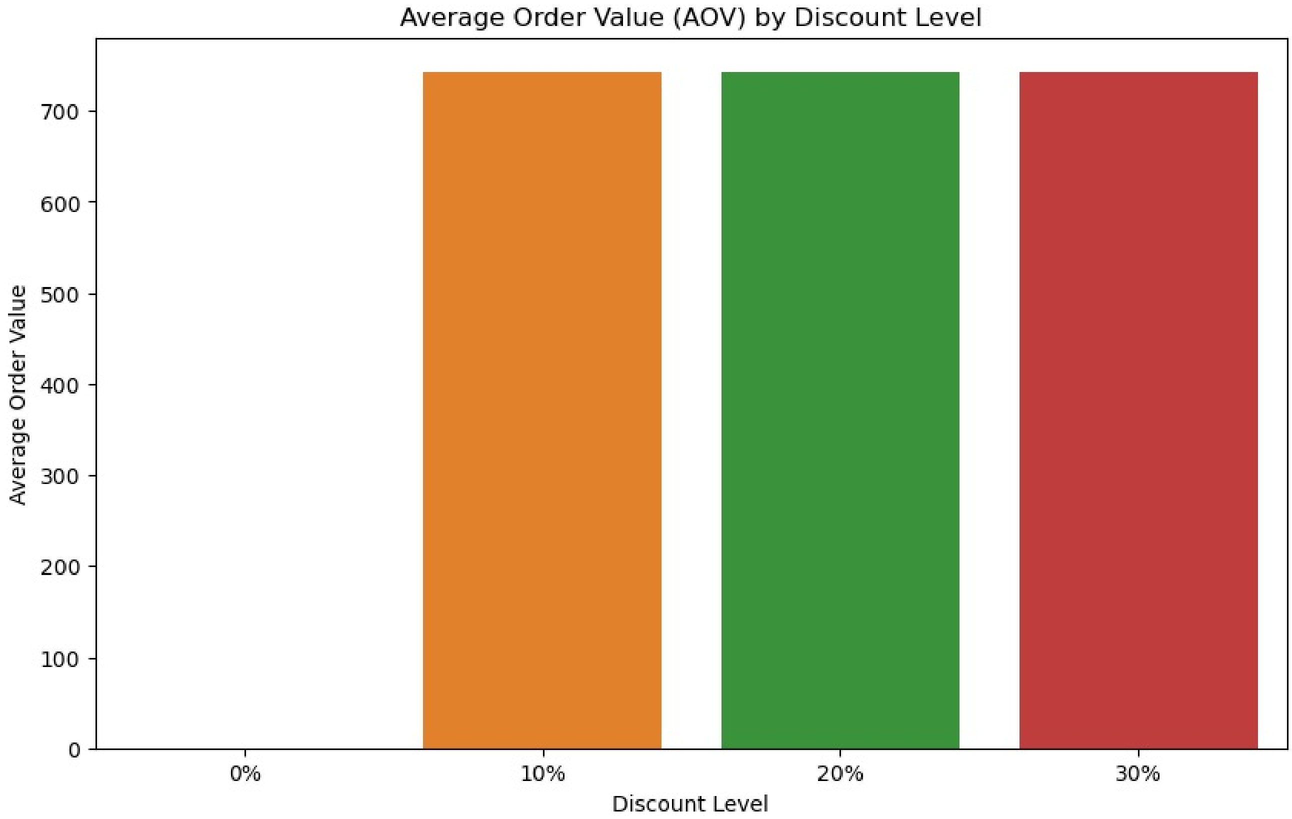
Discount Analysis: Investigate how discounts and promotions

affect revenue and customer engagement. Analyze KPIs like average order value and customer acquisition cost across different discount structures.

In [107…

```
# Assuming datasets are loaded: customer_df, taxamount_df, marketing_df, discount_df, onlinesales_df

# Merge discount information with sales data
sales_data = pd.merge(onlinesales_df, discount_df, on='Product_Category', how='left')

# Calculate Average Order Value (AOV)
sales_data['Order_Amount'] = sales_data['Quantity'] * sales_data['Avg_Price']
aov = sales_data.groupby('Transaction_ID')['Order_Amount'].sum().mean()

# Calculate Customer Acquisition Cost (CAC) - Example: simplified calculation
# CAC = Total Marketing Costs / Number of New Customers
total_marketing_costs = marketing_df['Online_Spend'].sum() + marketing_df['Offline_Spend'].sum()
new_customers = len(customer_df)

cac = total_marketing_costs / new_customers

# Calculate Total Revenue
total_revenue = sales_data['Order_Amount'].sum()

# Analyze Revenue Impact of Discounts
discounted_revenue = sales_data[sales_data['Discount_pct'] > 0]['Order_Amount'].sum()
non_discounted_revenue = sales_data[sales_data['Discount_pct'] == 0]['Order_Amount'].sum()

# Visualize the impact of discounts on revenue
plt.figure(figsize=(10, 6))
sns.barplot(x=['Discounted', 'Non-Discounted'], y=[discounted_revenue, non_discounted_revenue])
plt.title('Impact of Discounts on Revenue')
plt.xlabel('Discount Type')
plt.ylabel('Total Revenue')
plt.show()

# Perform Discount Sensitivity Analysis (Example: comparing AOV across different discount levels)
discount_levels = [0, 0.1, 0.2, 0.3]  # Example discount levels (0%, 10%, 20%, 30%)

aov_by_discount = []
for discount in discount_levels:
    temp_data = sales_data[sales_data['Discount_pct'] == discount]
    aov_by_discount.append(temp_data.groupby('Transaction_ID')['Order_Amount'].sum().mean())

# Visualize AOV by discount level
plt.figure(figsize=(10, 6))
sns.barplot(x=[f'{int(discount*100)}%' for discount in discount_levels], y=aov_by_discount)
plt.title('Average Order Value (AOV) by Discount Level')
plt.xlabel('Discount Level')
plt.ylabel('Average Order Value')
plt.show()

# Statistical Testing (if applicable)
# Example: Conduct t-test to compare AOV between different discount levels
from scipy.stats import ttest_ind

# Example: Comparing AOV between 0% discount and 20% discount
aov_0_percent = sales_data[sales_data['Discount_pct'] == 0]['Order_Amount']
aov_20_percent = sales_data[sales_data['Discount_pct'] == 0.2]['Order_Amount']

t_stat, p_value = ttest_ind(aov_0_percent, aov_20_percent)

if p_value < 0.05:
    print(f"Reject the null hypothesis: There is a significant difference in AOV between 0% and 20% discount lev
else:
    print("Fail to reject the null hypothesis.")
```






```
Fail to reject the null hypothesis.
```

Deeper Analysis:

Seasonality & Trends: Identify seasonal trends and patterns in sales data across different timeframes (month, week, day) to inform future marketing strategies.

In [64]:
```
onlinesales_df['Month'] = onlinesales_df['Transaction_Date'].dt.month
monthly_sales =  onlinesales_df.groupby(['Month']).size()
monthly_sales
```

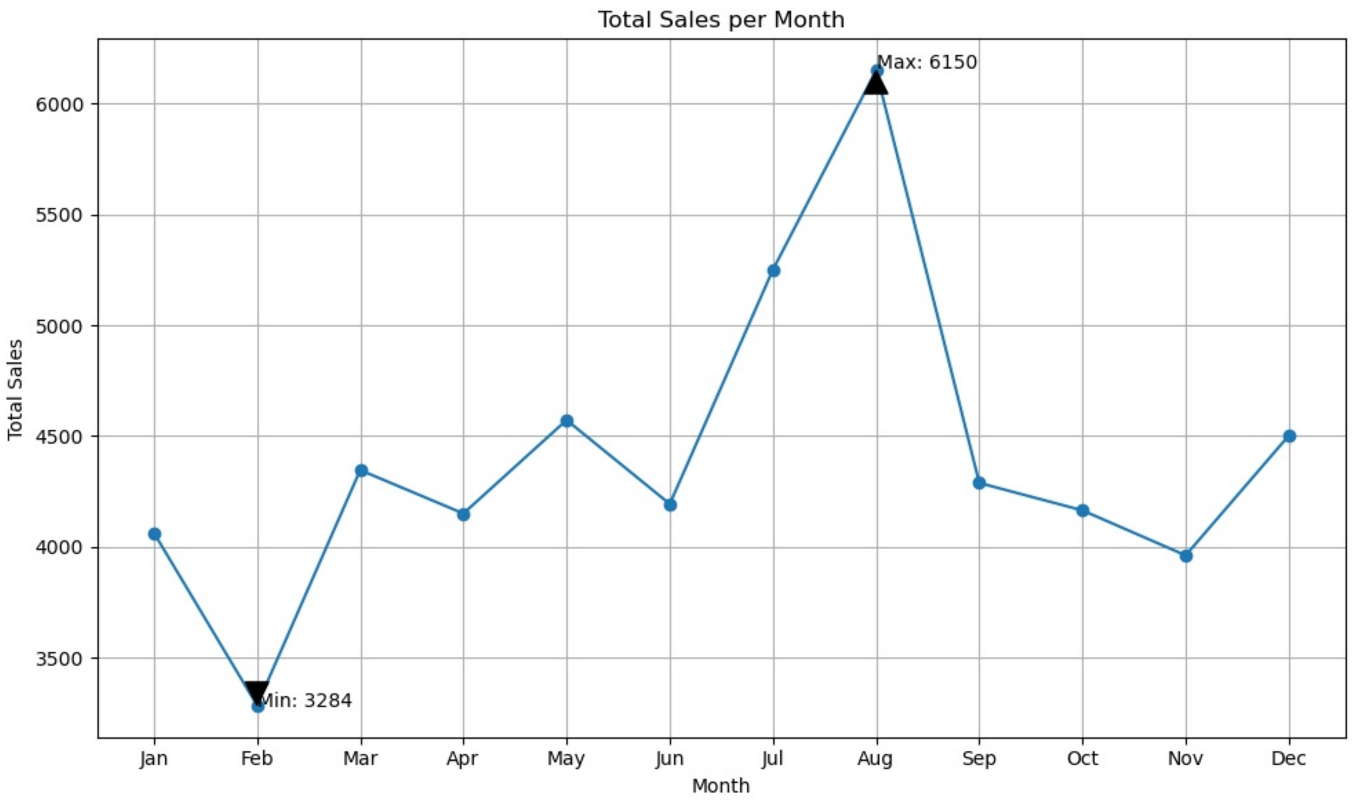
Out[64]:
```
Month
1     4063
2     3284
3     4346
4     4150
5     4572
6     4193
7     5251
8     6150
9     4288
10    4164
11    3961
12    4502
dtype: int64
```

In [65]:
```
total_sales_per_month = monthly_sales.groupby('Month').sum()
total_sales_per_month
```

Out[65]:
```
Month
1     4063
2     3284
3     4346
4     4150
5     4572
6     4193
7     5251
8     6150
9     4288
10    4164
11    3961
12    4502
dtype: int64
```

Calculate key performance indicators (KPIs) like revenue, number of orders, and average order value across various dimensions (category, month, week, day).

In [74]:
```
onlinesales_df['Month'] = onlinesales_df['Transaction_Date'].dt.month
monthly_sales = onlinesales_df.groupby([ 'Month']).size()
total_sales_per_month = monthly_sales.groupby('Month').sum()
plt.figure(figsize=(10, 6))
plt.plot(total_sales_per_month.index, total_sales_per_month.values, marker='o', linestyle='-')

# Retrieve and annotate the maximum value on the graph
max_value = total_sales_per_month.max()
max_month = total_sales_per_month.idxmax()
plt.annotate(f'Max: {max_value}', xy=(max_month, max_value), xytext=(max_month, max_value + 10),
             arrowprops=dict(facecolor='black', shrink=0.05),)

min_value = total_sales_per_month.min()
min_month = total_sales_per_month.idxmin()
plt.annotate(f'Min: {min_value}', xy=(min_month, min_value), xytext=(min_month, min_value - 10),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('Total Sales per Month')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(True)
plt.tight_layout()
plt.show()
```

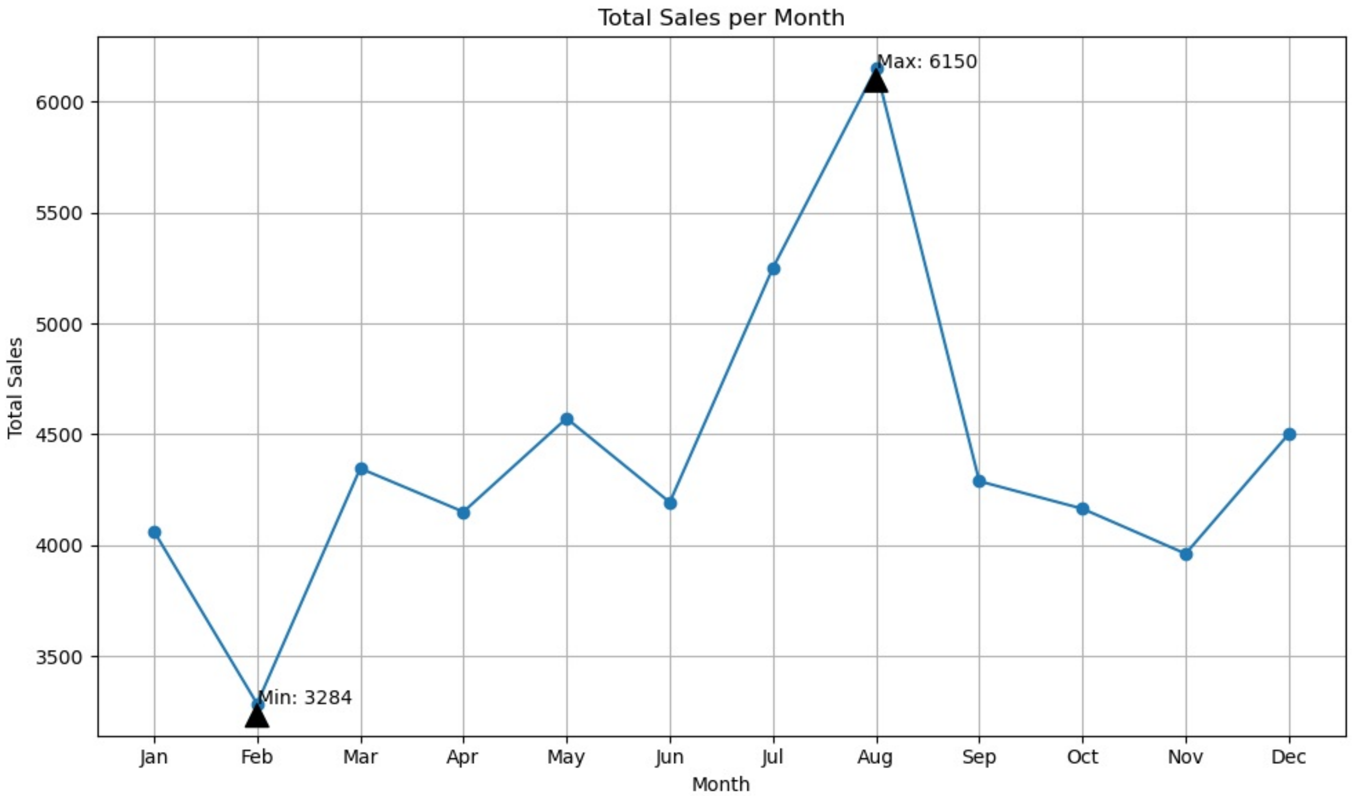


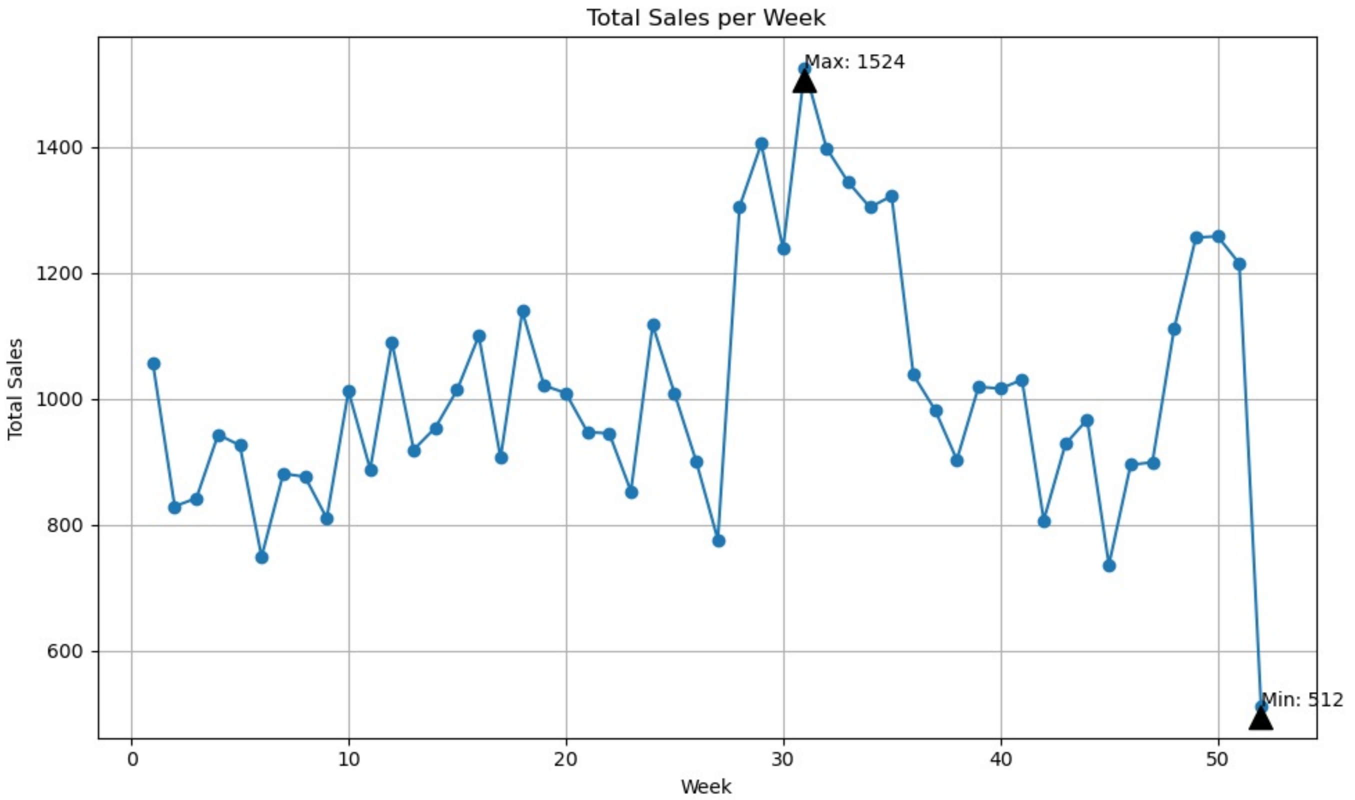
In [ ]:
```
In the graph we can see that Max sale was in month of August that is 6150 and minimum sale was in month of feb
```

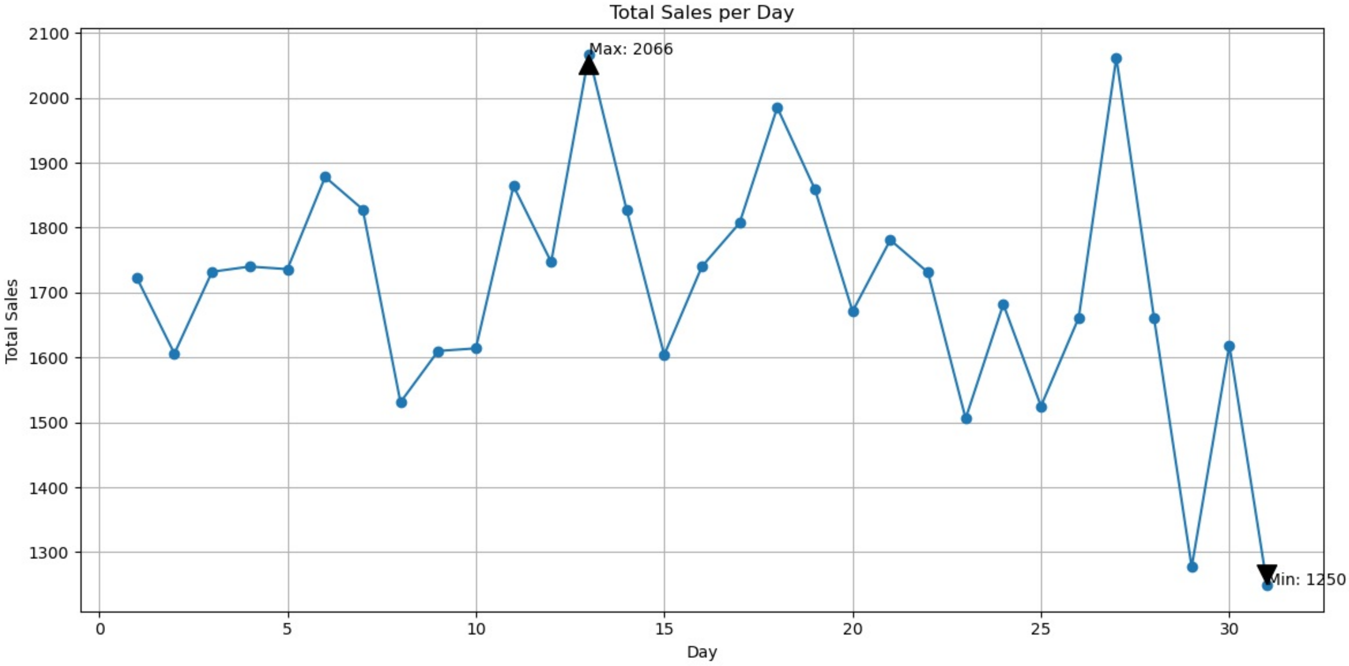
In [82]:
```
onlinesales_df = pd.read_csv(r'C:\\Users\\sinchan\\Block 2 project\\Online_Sales.csv')
```

In [83]:
```
onlinesales_df['Transaction_Date'] = pd.to_datetime(onlinesales_df['Transaction_Date'])

# Extract month, week, and day from Transaction_Date
onlinesales_df['Month'] = onlinesales_df['Transaction_Date'].dt.month
onlinesales_df['Week'] = onlinesales_df['Transaction_Date'].dt.isocalendar().week
onlinesales_df['Day'] = onlinesales_df['Transaction_Date'].dt.day

# Monthly Sales
monthly_sales = onlinesales_df.groupby('Month').size()
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', linestyle='-')
max_value = monthly_sales.max()
max_month = monthly_sales.idxmax()
plt.annotate(f'Max: {max_value}', xy=(max_month, max_value), xytext=(max_month, max_value + 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
min_value = monthly_sales.min()
min_month = monthly_sales.idxmin()
plt.annotate(f'Min: {min_value}', xy=(min_month, min_value), xytext=(min_month, min_value - 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('Total Sales per Month')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(True)
plt.tight_layout()
plt.show()

# Weekly Sales
weekly_sales = onlinesales_df.groupby('Week').size()
plt.figure(figsize=(10, 6))
plt.plot(weekly_sales.index, weekly_sales.values, marker='o', linestyle='-')
max_value = weekly_sales.max()
max_week = weekly_sales.idxmax()
plt.annotate(f'Max: {max_value}', xy=(max_week, max_value), xytext=(max_week, max_value + 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
min_value = weekly_sales.min()
min_week = weekly_sales.idxmin()
plt.annotate(f'Min: {min_value}', xy=(min_week, min_value), xytext=(min_week, min_value - 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
plt.xlabel('Week')
plt.ylabel('Total Sales')
plt.title('Total Sales per Week')
plt.grid(True)
plt.tight_layout()
plt.show()

# Daily Sales
```

```
daily_sales = onlinesales_df.groupby('Day').size()
plt.figure(figsize=(12, 6))
plt.plot(daily_sales.index, daily_sales.values, marker='o', linestyle='-')
max_value = daily_sales.max()
max_day = daily_sales.idxmax()
plt.annotate(f'Max: {max_value}', xy=(max_day, max_value), xytext=(max_day, max_value + 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
min_value = daily_sales.min()
min_day = daily_sales.idxmin()
plt.annotate(f'Min: {min_value}', xy=(min_day, min_value), xytext=(min_day, min_value - 1),
             arrowprops=dict(facecolor='black', shrink=0.05))
plt.xlabel('Day')
plt.ylabel('Total Sales')
plt.title('Total Sales per Day')
plt.grid(True)
plt.tight_layout()
plt.show()
```





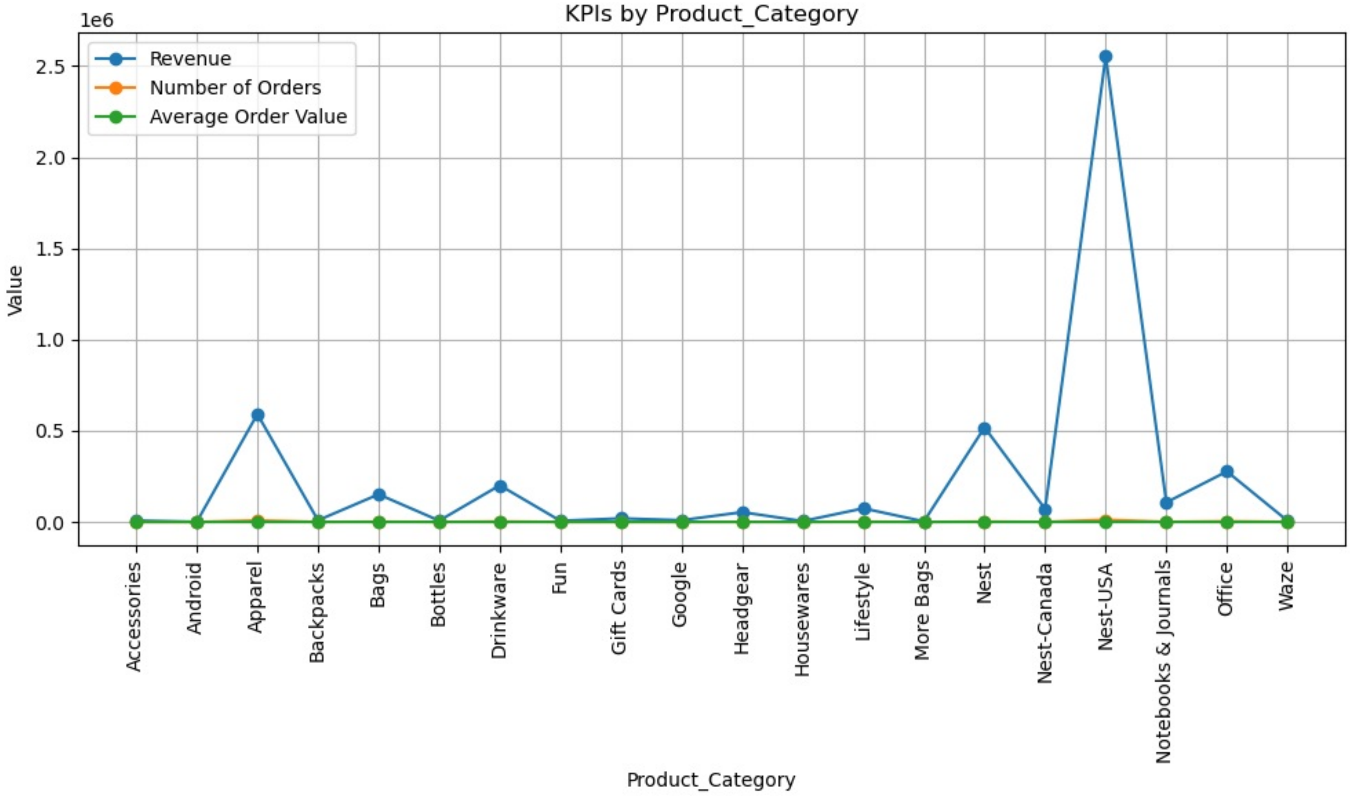


In [ ]:

In [112…
```
# Calculate Revenue
onlinesales_df['Revenue'] = onlinesales_df['Quantity'] * onlinesales_df['Avg_Price']

# Calculate KPIs across different dimensions
dimensions = ['Product_Category', 'Month', 'Week', 'Day']

for dimension in dimensions:
    grouped_df = onlinesales_df.groupby(dimension).agg(
        Revenue=('Revenue', 'sum'),
        Number_of_Orders=('Transaction_ID', 'nunique'),
        Average_Order_Value=('Revenue', 'mean')
    ).reset_index()

    print(f"KPIs by {dimension}:")
    print(grouped_df)
    print("\n")

    # Plotting
    plt.figure(figsize=(10, 6))
    plt.plot(grouped_df[dimension], grouped_df['Revenue'], marker='o', linestyle='-', label='Revenue')
    plt.plot(grouped_df[dimension], grouped_df['Number_of_Orders'], marker='o', linestyle='-', label='Number of
    plt.plot(grouped_df[dimension], grouped_df['Average_Order_Value'], marker='o', linestyle='-', label='Average
    plt.xlabel(dimension)
    plt.xticks(rotation=90)
    plt.ylabel('Value')
    plt.title(f'KPIs by {dimension}')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

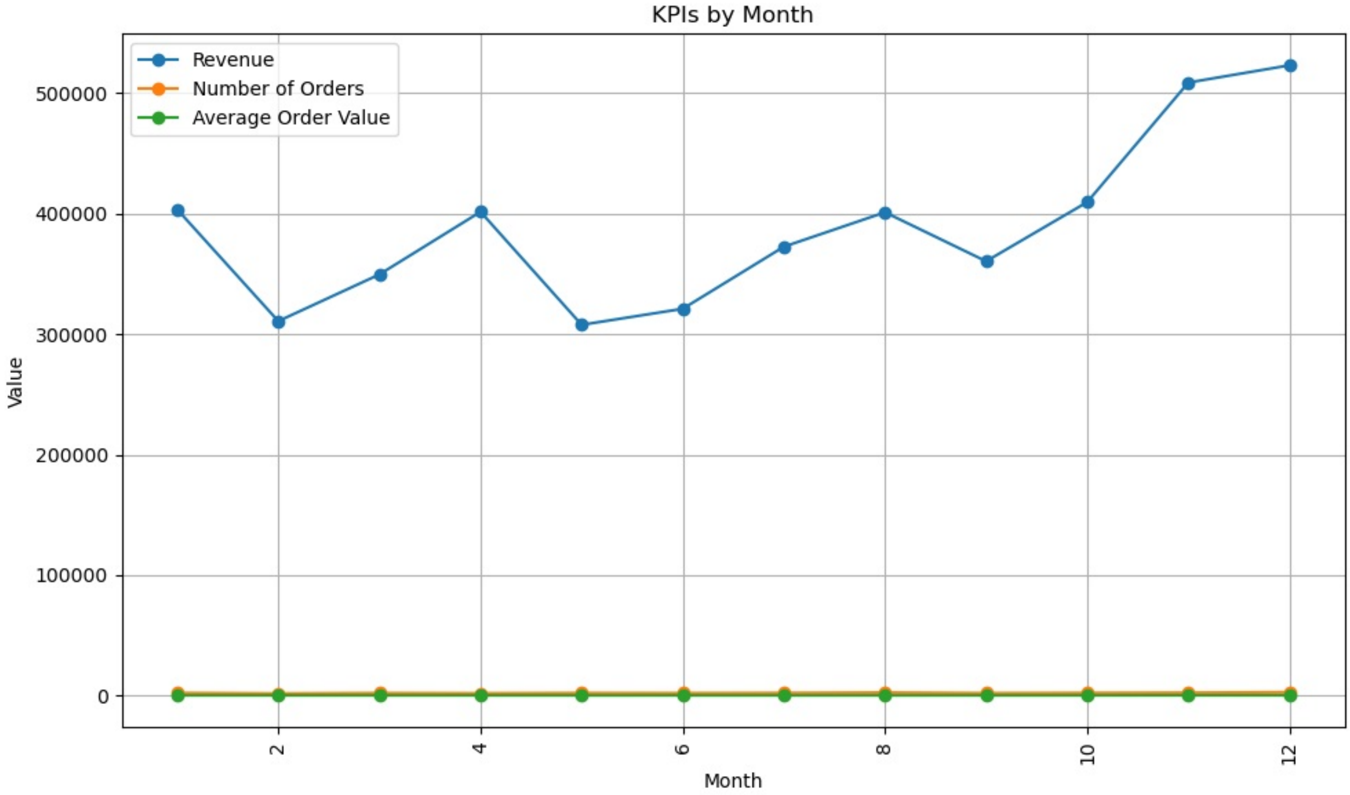
KPIs by Product_Category:

	Product_Category	Revenue	Number_of_Orders	Average_Order_Value
0	Accessories	7295.88	191	31.178974
1	Android	711.03	43	16.535581
2	Apparel	591145.80	8129	32.613141
3	Backpacks	8772.69	84	98.569551
4	Bags	151314.43	1545	80.400866
5	Bottles	6923.65	258	25.834515
6	Drinkware	200707.83	2524	57.624987
7	Fun	6029.01	146	37.681313
8	Gift Cards	19533.82	157	122.854214
9	Google	9420.47	105	89.718762
10	Headgear	53471.44	674	69.353359
11	Housewares	4637.32	122	38.010820
12	Lifestyle	74385.70	1712	24.057471
13	More Bags	2946.96	40	64.064348
14	Nest	518193.50	1974	235.756824
15	Nest-Canada	70910.40	258	223.692114
16	Nest-USA	2554202.39	11626	182.273774
17	Notebooks & Journals	107085.96	620	142.971909
18	Office	276794.40	3526	42.498756
19	Waze	6311.94	442	11.393394



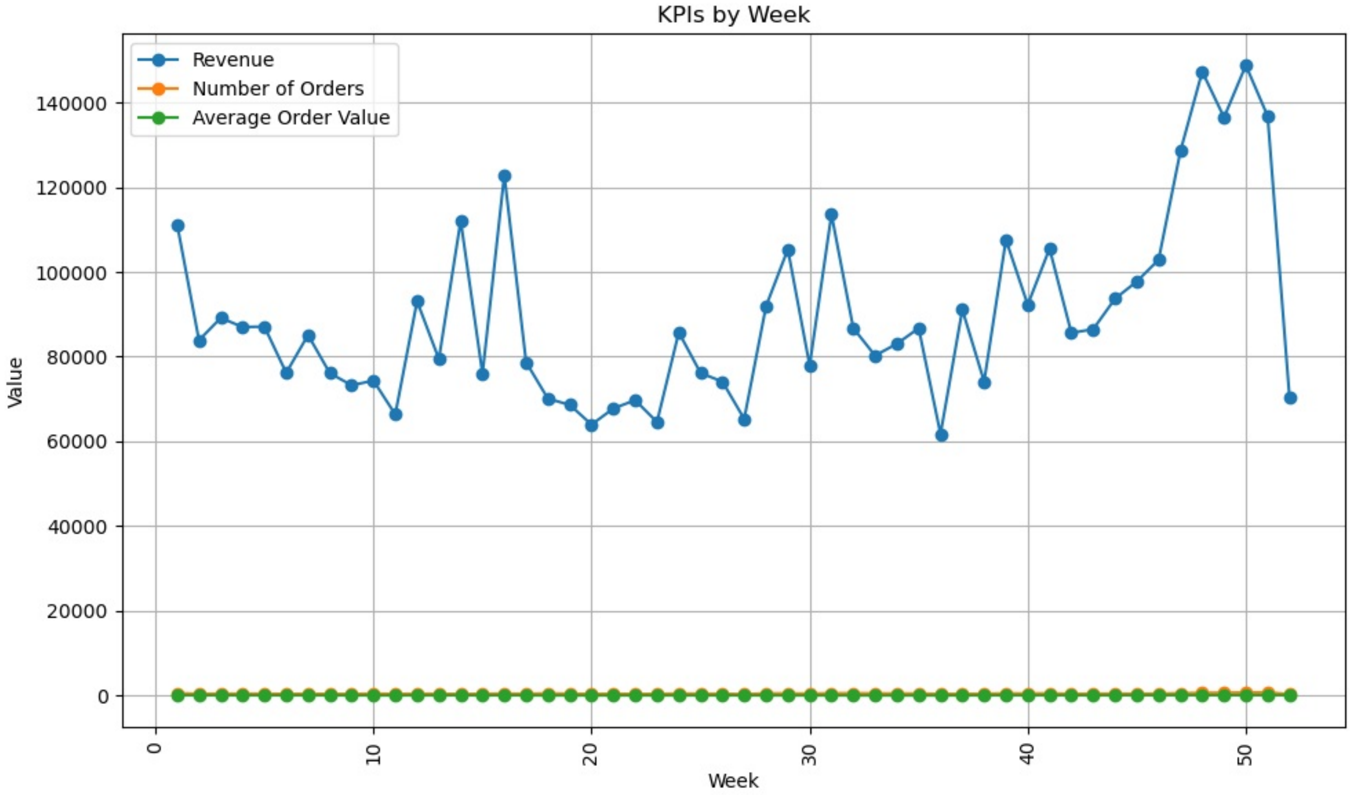
KPIs by Month:

	Month	Revenue	Number_of_Orders	Average_Order_Value
0	1	403624.58	2102	99.341516
1	2	310819.80	1664	94.646711
2	3	349608.09	1991	80.443647
3	4	401618.42	1813	96.775523
4	5	307763.42	2034	67.314834
5	6	321081.38	1940	76.575574
6	7	372638.07	2080	70.965163
7	8	401210.37	2414	65.237459
8	9	360548.40	1932	84.083116
9	10	409681.28	2125	98.386475
10	11	508942.62	2282	128.488417
11	12	523258.19	2684	116.227941


KPIs by Month
Revenue
Number of Orders
Average Order Value
Value
500000
400000
300000
200000
100000
0
Month
2
4
6
8
10
12

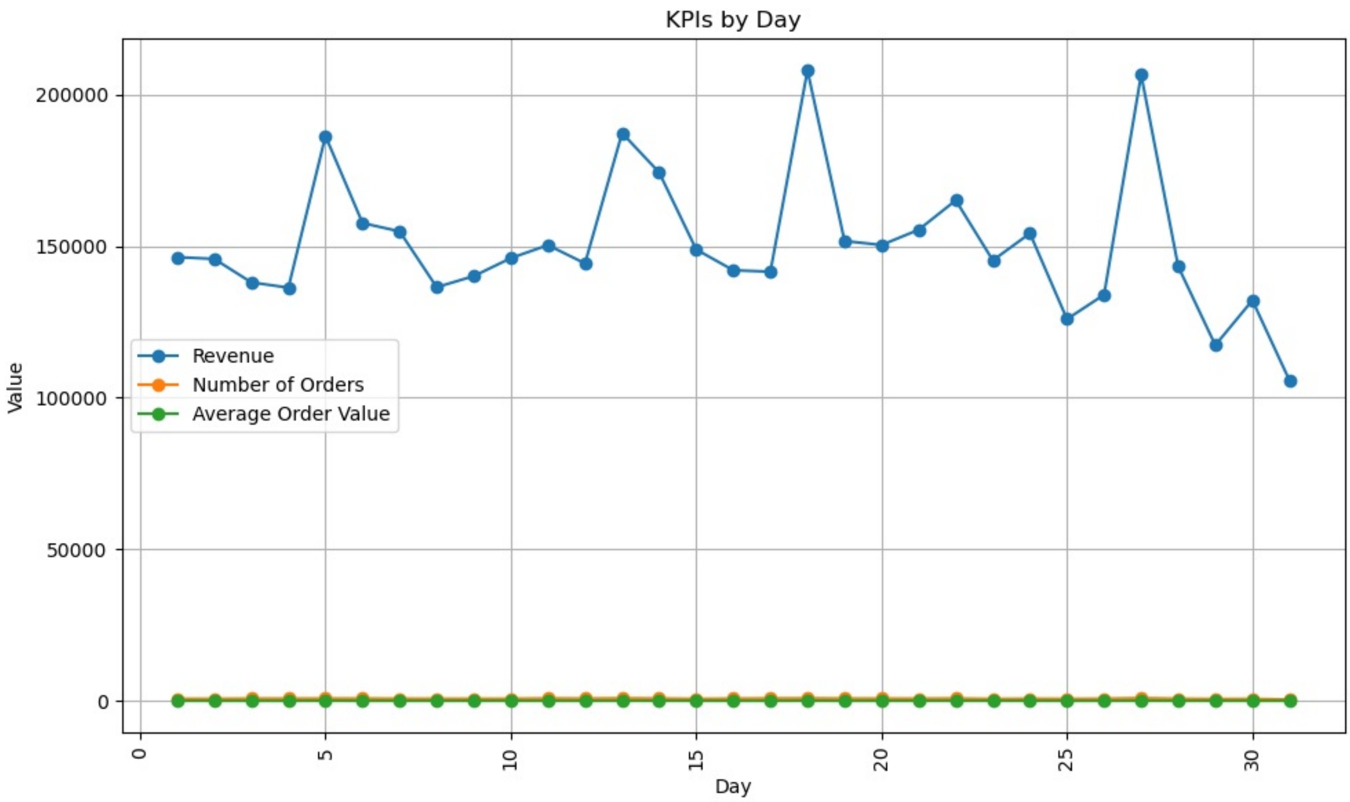
KPIs by Week:

	Week	Revenue	Number_of_Orders	Average_Order_Value
0	1	111197.29	573	105.300464
1	2	83877.67	428	101.179337
2	3	89160.36	466	105.891164
3	4	87005.26	461	92.264327
4	5	87101.98	463	94.062613
5	6	76181.43	403	101.710854
6	7	85120.50	454	96.618048
7	8	76107.07	405	86.880217
8	9	73253.25	407	90.436111
9	10	74351.00	478	73.469368
10	11	66372.83	409	74.744178
11	12	93099.97	455	85.412817
12	13	79440.65	444	86.442492
13	14	112119.14	376	117.525304
14	15	75948.03	440	74.899438
15	16	122924.99	480	111.749991
16	17	78673.55	433	86.836148
17	18	70095.44	534	61.541212
18	19	68628.37	451	67.216817
19	20	63974.32	423	63.403687
20	21	67781.49	439	71.574963
21	22	69740.00	405	73.798942
22	23	64579.29	412	75.708429
23	24	85747.70	495	76.766070
24	25	76136.63	462	75.457512
25	26	74114.27	437	82.257791
26	27	65317.18	366	84.171624
27	28	91918.86	525	70.435908
28	29	105348.14	527	74.980883
29	30	77926.99	466	62.945872
30	31	113621.52	561	74.554803
31	32	86746.69	564	62.050565
32	33	80266.73	532	59.722269
33	34	83054.80	509	63.692331
34	35	86718.78	502	65.596657
35	36	61729.43	419	59.469586
36	37	91109.27	430	92.779297
37	38	74140.64	447	82.104806
38	39	107616.20	528	105.609617
39	40	92255.98	502	90.803130
40	41	105533.34	536	102.459553
41	42	85621.96	428	106.099083
42	43	86478.24	461	93.187759
43	44	93680.01	469	96.876949
44	45	97751.72	451	132.814837
45	46	102883.24	474	114.953341
46	47	128513.53	557	142.951646
47	48	147300.25	658	132.583483
48	49	136485.72	687	108.666975
49	50	148775.29	726	118.263347
50	51	136787.27	756	112.582115
51	52	70480.36	347	137.656953



KPIs by Day:

	Day	Revenue	Number_of_Orders	Average_Order_Value
0	1	146340.61	779	84.933610
1	2	145764.64	759	90.762540
2	3	138027.51	834	79.692558
3	4	136286.33	831	78.325477
4	5	186275.66	845	107.301647
5	6	157578.08	849	83.907391
6	7	154841.36	806	84.705339
7	8	136434.08	782	89.114357
8	9	140098.13	773	87.017472
9	10	146020.67	775	90.471295
10	11	150309.33	890	80.638053
11	12	144279.87	839	82.587218
12	13	187253.48	926	90.635760
13	14	174261.00	858	95.328775
14	15	148856.99	734	92.803610
15	16	142020.40	825	81.620920
16	17	141550.55	884	78.334560
17	18	207920.05	912	104.745617
18	19	151653.66	860	81.578085
19	20	150354.20	828	89.978576
20	21	155379.45	817	87.242813
21	22	165075.28	873	95.364113
22	23	145278.27	723	96.466315
23	24	154192.94	793	91.672378
24	25	125907.94	734	82.562584
25	26	133956.81	793	80.648290
26	27	206501.11	980	100.146028
27	28	143216.74	782	86.223203
28	29	117328.37	698	91.878128
29	30	132112.94	727	81.601569
30	31	105718.17	552	84.574536



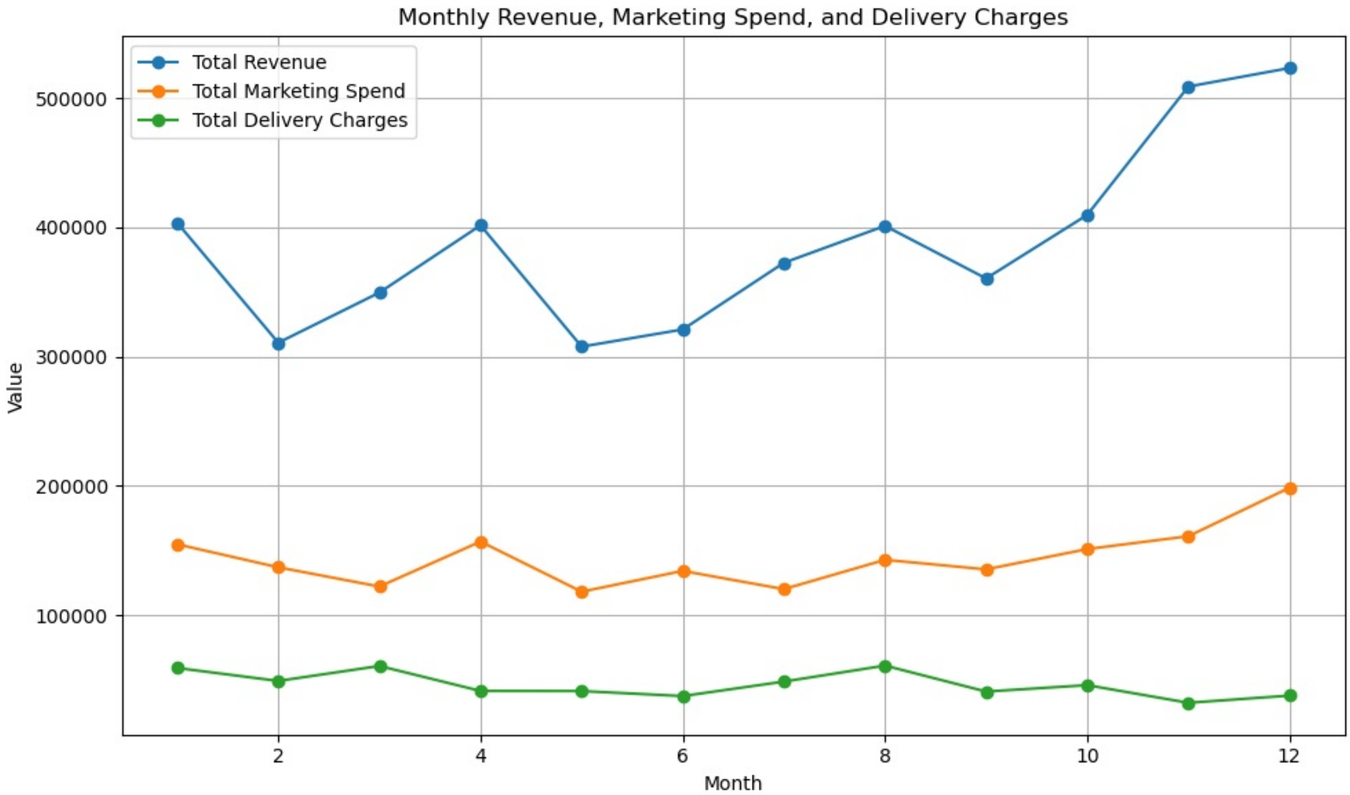
Calculate revenue, marketing spend, and delivery charges by month to understand their correlation. This can reveal areas for optimization.

In [90]:
```
marketing_df = pd.read_csv(r'C:\Users\sinchan\Block 2 project\Marketing_Spend.csv')
```

In [92]:
```
onlinesales_df['Revenue'] = onlinesales_df['Quantity'] * onlinesales_df['Avg_Price']

# Extract month from Transaction_Date for grouping
onlinesales_df['Month'] = onlinesales_df['Transaction_Date'].dt.month

# Group by month to calculate total revenue and delivery charges
monthly_sales = onlinesales_df.groupby('Month').agg(
    Total_Revenue=('Revenue', 'sum'),
    Total_Delivery_Charges=('Delivery_Charges', 'sum')
).reset_index()

# Group by month to calculate total revenue and delivery charges
marketing_df['Date'] = pd.to_datetime(marketing_df['Date'])

# Make sure you have executed the code to create the 'Month' column in onlinesales_df
onlinesales_df['Transaction_Date'] = pd.to_datetime(onlinesales_df['Transaction_Date'])
onlinesales_df['Month'] = onlinesales_df['Transaction_Date'].dt.month

monthly_sales = onlinesales_df.groupby('Month').agg(
    Total_Revenue=('Revenue', 'sum'),
    Total_Delivery_Charges=('Delivery_Charges', 'sum')
).reset_index()

# Extract month from Date in marketing_df for merging
marketing_df['Month'] = marketing_df['Date'].dt.month

# Group marketing data by month and calculate total marketing spend
monthly_marketing_spend = marketing_df.groupby('Month').agg(
    Total_Offline_Spend=('Offline_Spend', 'sum'),
    Total_Online_Spend=('Online_Spend', 'sum')
).reset_index()

# Merge monthly_sales with monthly_marketing_spend on 'Month'
monthly_data = pd.merge(monthly_sales, monthly_marketing_spend, on='Month', how='left')

# Calculate Total_Marketing_Spend
monthly_data['Total_Marketing_Spend'] = monthly_data['Total_Offline_Spend'] + monthly_data['Total_Online_Spend'

# Display the monthly data
print("Monthly Data:")
print(monthly_data)
```

```python
# Calculate correlation matrix
correlation_matrix = monthly_data[['Total_Revenue', 'Total_Marketing_Spend', 'Total_Delivery_Charges']].corr()
print("\nCorrelation Matrix:")
print(correlation_matrix)

# Plotting the data
plt.figure(figsize=(10, 6))
plt.plot(monthly_data['Month'], monthly_data['Total_Revenue'], marker='o', linestyle='-', label='Total Revenue'
plt.plot(monthly_data['Month'], monthly_data['Total_Marketing_Spend'], marker='o', linestyle='-', label='Total I
plt.plot(monthly_data['Month'], monthly_data['Total_Delivery_Charges'], marker='o', linestyle='-', label='Total
plt.xlabel('Month')
plt.ylabel('Value')
plt.title('Monthly Revenue, Marketing Spend, and Delivery Charges')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

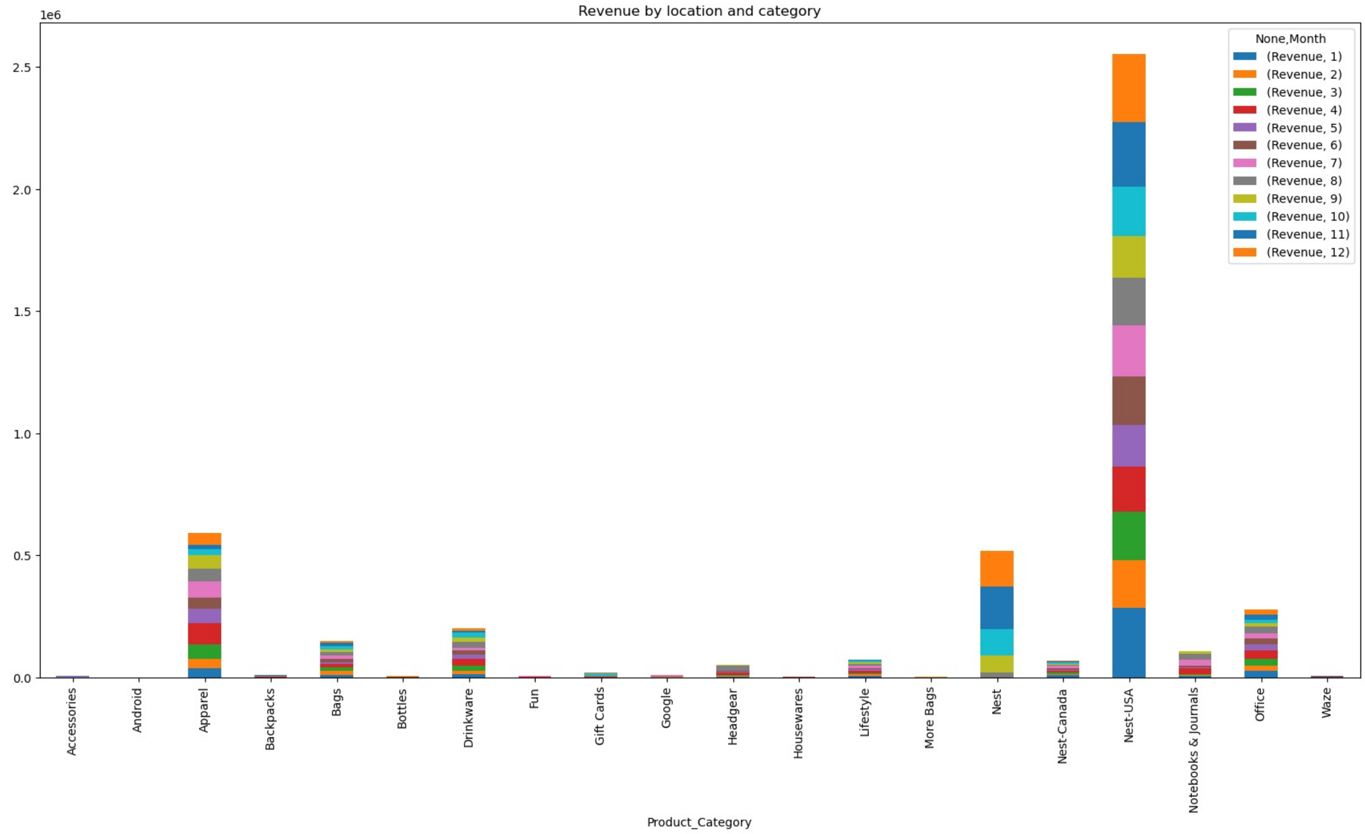
```
Monthly Data:
    Month  Total_Revenue  Total_Delivery_Charges  Total_Offline_Spend  \
0       1      403624.58                59242.32                96600
1       2      310819.80                49216.60                81300
2       3      349608.09                60799.94                73500
3       4      401618.42                41481.74                96000
4       5      307763.42                41396.17                65500
5       6      321081.38                37513.58                80500
6       7      372638.07                48723.93                67500
7       8      401210.37                61099.57                85500
8       9      360548.40                41005.42                83000
9      10      409681.28                45961.88                93500
10     11      508942.62                32311.93                93000
11     12      523258.19                37881.99               122000

    Total_Online_Spend  Total_Marketing_Spend
0             58328.95              154928.95
1             55807.92              137107.92
2             48750.09              122250.09
3             61026.83              157026.83
4             52759.64              118259.64
5             53818.14              134318.14
6             52717.85              120217.85
7             57404.15              142904.15
8             52514.54              135514.54
9             57724.65              151224.65
10            68144.96              161144.96
11            76648.75              198648.75

Correlation Matrix:
                        Total_Revenue  Total_Marketing_Spend  \
Total_Revenue                1.000000               0.851503
Total_Marketing_Spend        0.851503               1.000000
Total_Delivery_Charges      -0.296728              -0.325481

                        Total_Delivery_Charges
Total_Revenue                        -0.296728
Total_Marketing_Spend                -0.325481
Total_Delivery_Charges                1.000000
```



In [ ]:

In [104…

```
onlinesales_df.groupby(by=['Product_Category','Month'])[['Revenue']].sum().unstack().plot(kind='bar',figsize=[2(
plt.show()
```



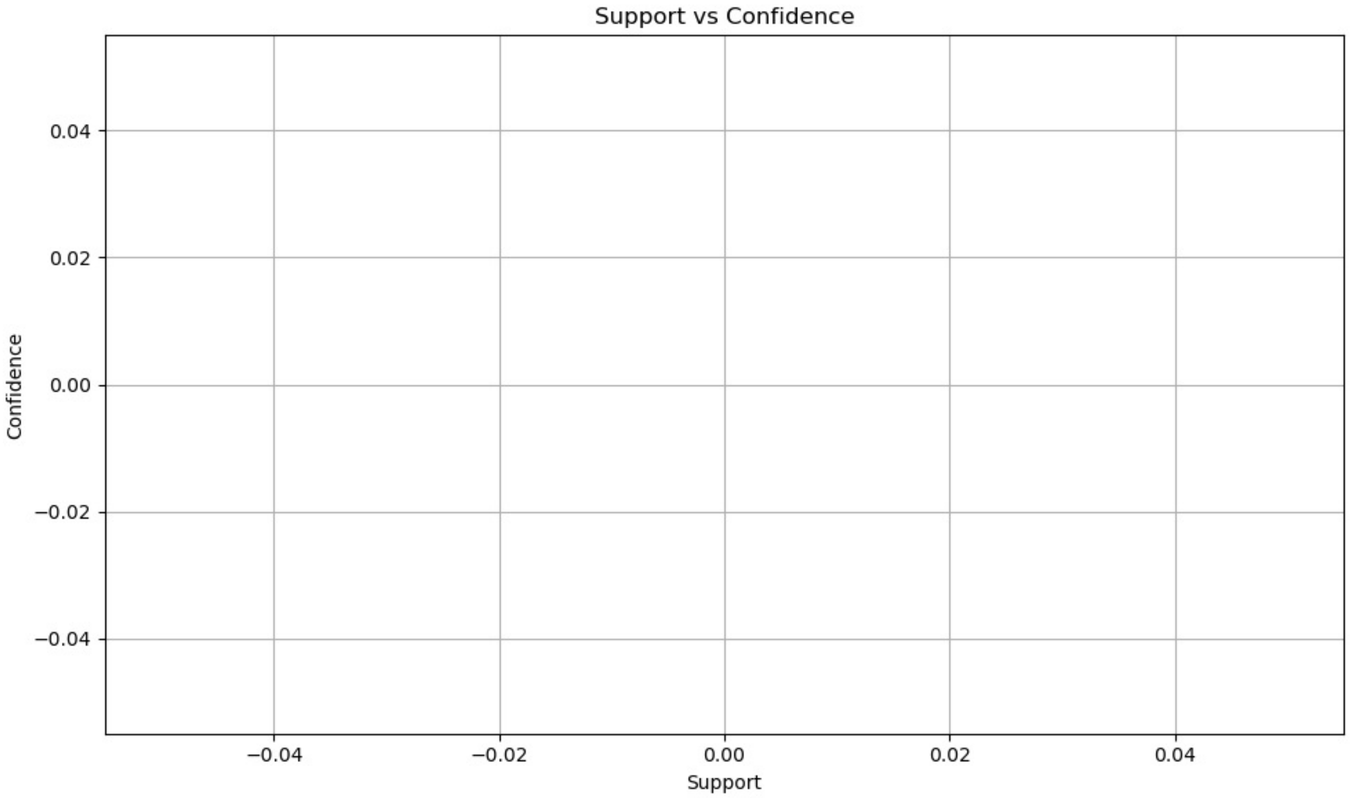
Analyze co-purchased products through market basket analysis. This will uncover cross-selling opportunities and inform product placement strategies.

In [108…

```
from mlxtend.frequent_patterns import apriori, association_rules

# Create a basket format where rows are transactions and columns are products
basket = (onlinesales_df.groupby(['Transaction_ID', 'Product_SKU'])['Product_SKU']
          .count().unstack().reset_index().fillna(0).set_index('Transaction_ID'))

# Convert values to binary (0 or 1) for presence of products
basket = basket.applymap(lambda x: 1 if x > 0 else 0)

# Apply Apriori algorithm to find frequent itemsets
# Lowering min_support to find some frequent itemsets
frequent_itemsets = apriori(basket, min_support=0.02, use_colnames=True)  # Lowered min_support

# Generate association rules
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)

# Filter rules based on a minimum confidence level
# Lowering confidence threshold to get some rules
filtered_rules = rules[rules['confidence'] >= 0.3]  # Lowered confidence threshold

# Display the rules
print("Association Rules:")
print(filtered_rules)

# Plot the rules
plt.figure(figsize=(10, 6))
plt.scatter(filtered_rules['support'], filtered_rules['confidence'], alpha=0.5, marker="o")
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Support vs Confidence')
plt.grid(True)
plt.tight_layout()
plt.show()

# Generate a heatmap of the lift values
# Correctly using the pivot method to create a pivot table for the heatmap
# Handling the potential for an empty DataFrame after filtering
if not filtered_rules.empty:
    pivot_table = filtered_rules.pivot(index='antecedents', columns='consequents', values='lift')
    plt.figure(figsize=(10, 6))
    sns.heatmap(pivot_table, annot=True, cmap="YlGnBu", cbar=True)
    plt.title('Lift Heatmap of Association Rules')
    plt.tight_layout()
```

```
    plt.show()
else:
    print("No association rules meet the filtering criteria.")
```

```
Association Rules:
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, leverage, conviction, zhangs_metric]
Index: []
```




```
No association rules meet the filtering criteria.
```

Customer Lifetime Value (CLTV): Implement predictive models to estimate the future value of each customer. This helps prioritize retention efforts for high-value customers. (Optional)

In [109…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Calculate RFM metrics
current_date = pd.to_datetime('2023-07-01')  # Assuming current date for recency calculation
rfm_df = onlinesales_df.groupby('CustomerID').agg({
    'Transaction_Date': lambda x: (current_date - x.max()).days,
    'Transaction_ID': 'count',
    'Revenue': 'sum'
}).rename(columns={'Transaction_Date': 'Recency', 'Transaction_ID': 'Frequency', 'Revenue': 'Monetary'})

# Prepare features and target variable
X = rfm_df[['Recency', 'Frequency', 'Monetary']]
y = rfm_df['Monetary']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')

# Predict CLTV for all customers
rfm_df['Predicted_CLTV'] = model.predict(rfm_df[['Recency', 'Frequency', 'Monetary']])
```

```
print("Customer Lifetime Value (CLTV) Predictions:")
print(rfm_df)
```

```
Mean Squared Error: 2.042345704249675e-24
R^2 Score: 1.0
Customer Lifetime Value (CLTV) Predictions:
            Recency  Frequency  Monetary  Predicted_CLTV
CustomerID
12346          1385          2     30.99           30.99
12347          1337         60  13834.90        13834.90
12348          1351         23   1442.12         1442.12
12350          1295         17   1360.07         1360.07
12356          1385         36   1442.47         1442.47
...             ...        ...       ...             ...
18259          1548          7    544.34          544.34
18260          1365         40   2363.05         2363.05
18269          1472          8    101.56          101.56
18277          1347          1    298.00          298.00
18283          1360        102   6362.77         6362.77

[1468 rows x 4 columns]
```

In [ ]:
```
Identify Customer Acquisition Month
We need to find the first purchase date for each customer to determine their acquisition month.
```

In [116…
```
onlinesales_df['Acquisition_Month'] = onlinesales_df.groupby('CustomerID')['Transaction_Date'].transform('min')
```

In [ ]:
```
Create Monthly Cohorts
Now, we'll create cohorts based on the acquisition month and track their behavior over time
```

In [117…
```
onlinesales_df['Transaction_Month'] = onlinesales_df['Transaction_Date'].dt.to_period('M')
cohort_data = onlinesales_df.groupby(['Acquisition_Month', 'Transaction_Month']).agg({
    'CustomerID': 'nunique',
    'Revenue': 'sum'
}).reset_index()
cohort_data.rename(columns={'CustomerID': 'Total_Customers'}, inplace=True)
```

In [ ]:
```
Calculate Retention Rate
To calculate the retention rate, we need the number of customers who made a purchase in each subsequent month d
```

In [118…
```
cohort_pivot = cohort_data.pivot_table(index='Acquisition_Month', columns='Transaction_Month', values='Total_Cu
cohort_size = cohort_pivot.iloc[:, 0]
retention_rate = cohort_pivot.divide(cohort_size, axis=0)
```

In [ ]:
```
 Visualize the Cohort Analysis
We'll create a heatmap to visualize the retention rate of each cohort over time.
```

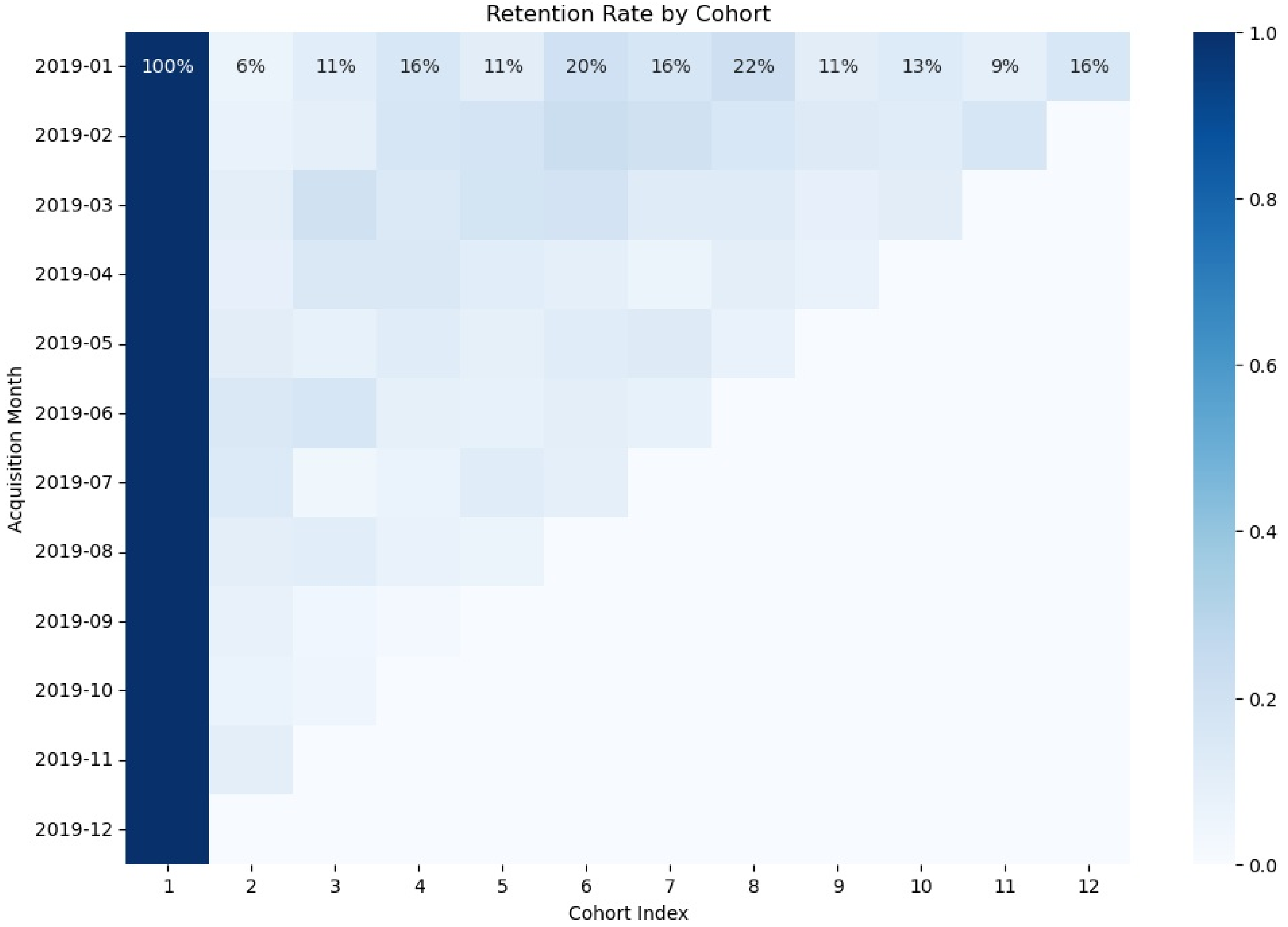
In [120…
```
plt.figure(figsize=(12, 8))
sns.heatmap(retention_rate, annot=True, fmt='.0%', cmap='YlGnBu')
plt.title('Customer Retention Rate by Monthly Cohorts')
plt.xlabel('Transaction Month')
plt.ylabel('Acquisition Month')
plt.show()
```

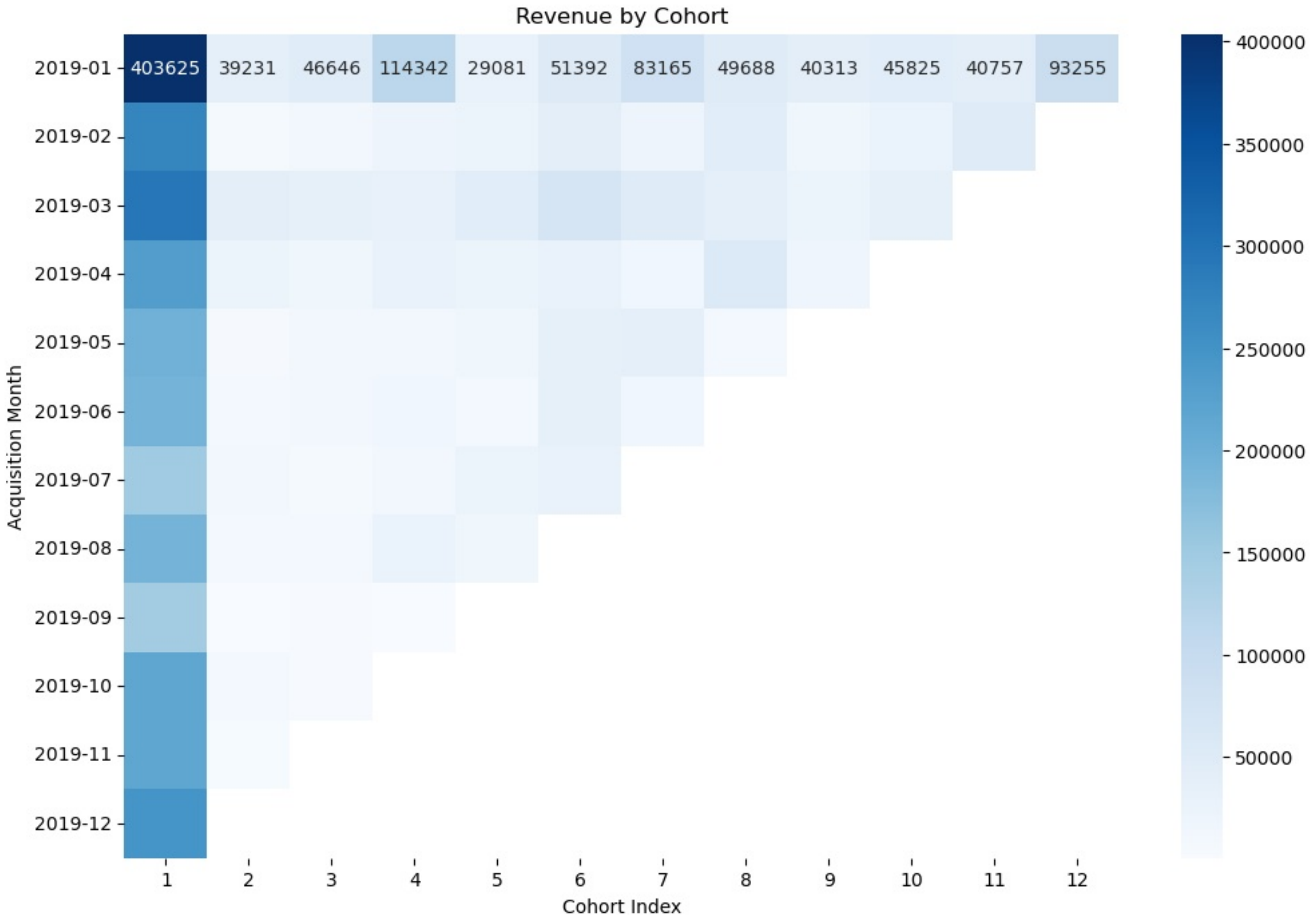
In [ ]:

In [115…

```
# Create the acquisition month
onlinesales_df['Acquisition_Month'] = onlinesales_df.groupby('CustomerID')['Transaction_Date'].transform('min')

# Create the cohort month
onlinesales_df['Cohort_Month'] = onlinesales_df['Transaction_Date'].dt.to_period('M')

# Calculate the difference in months between the transaction date and the acquisition date
onlinesales_df['Cohort_Index'] = (onlinesales_df['Cohort_Month'].dt.year - onlinesales_df['Acquisition_Month'].

# Aggregate data to get the cohort table
cohort_data = onlinesales_df.groupby(['Acquisition_Month', 'Cohort_Index']).agg(
    Customers=('CustomerID', 'nunique'),
    Orders=('Transaction_ID', 'nunique'),
    Total_Revenue=('Revenue', 'sum')
).reset_index()

# Pivot the cohort table to get retention matrix
cohort_counts = cohort_data.pivot_table(index='Acquisition_Month', columns='Cohort_Index', values='Customers')

# Fill the NaN values with 0
cohort_counts.fillna(0, inplace=True)

# Calculate retention rate
cohort_sizes = cohort_counts.iloc[:, 0]
retention_matrix = cohort_counts.divide(cohort_sizes, axis=0)

# Plot the retention matrix
plt.figure(figsize=(12, 8))
plt.title('Retention Rate by Cohort')
sns.heatmap(retention_matrix, annot=True, fmt=".0%", cmap="Blues")
plt.xlabel('Cohort Index')
plt.ylabel('Acquisition Month')
plt.show()

# Plot the revenue matrix
revenue_matrix = cohort_data.pivot_table(index='Acquisition_Month', columns='Cohort_Index', values='Total_Reven
plt.figure(figsize=(12, 8))
plt.title('Revenue by Cohort')
sns.heatmap(revenue_matrix, annot=True, fmt=".0f", cmap="Blues")
plt.xlabel('Cohort Index')
plt.ylabel('Acquisition Month')
plt.show()
```

```python
# Show the cohort table
cohort_table = cohort_data.pivot_table(index='Acquisition_Month', columns='Cohort_Index', values=['Orders', 'To
print(cohort_table)
```





```
                   Orders                                                \
Cohort_Index            1      2      3      4      5      6      7      8   
Acquisition_Month                                                            
2019-01            2102.0  218.0  294.0  353.0  216.0  355.0  400.0  337.0   
2019-02            1461.0   60.0   87.0  159.0  145.0  271.0  161.0  220.0   
2019-03            1676.0  167.0  233.0  163.0  262.0  352.0  270.0  186.0   
2019-04            1272.0  121.0  126.0  185.0  127.0  166.0   66.0  236.0   
2019-05            1379.0   55.0   89.0  104.0   92.0  223.0  165.0   67.0   
2019-06            1189.0   87.0   90.0  104.0   75.0  190.0   94.0    NaN   
2019-07             924.0  113.0   33.0   87.0   98.0  154.0    NaN    NaN   
2019-08            1299.0   59.0   66.0  132.0   78.0    NaN    NaN    NaN   
2019-09             840.0   13.0   19.0    4.0    NaN    NaN    NaN    NaN   
2019-10            1136.0   29.0   17.0    NaN    NaN    NaN    NaN    NaN   
2019-11            1044.0   21.0    NaN    NaN    NaN    NaN    NaN    NaN   
2019-12            1288.0    NaN    NaN    NaN    NaN    NaN    NaN    NaN   

                                 ... Total_Revenue                        \
Cohort_Index           9      10  ...             3          4         5   
Acquisition_Month                 ...                                      
2019-01            240.0  252.0  ...      46645.94  114342.41  29081.40   
2019-02             96.0  126.0  ...      12709.89   21995.59  23458.86   
2019-03            128.0  209.0  ...      36365.25   30603.93  46287.73   
2019-04             96.0    NaN  ...      16597.02   29717.78  25249.90   
2019-05              NaN    NaN  ...      11887.83   14110.62  17265.29   
2019-06              NaN    NaN  ...      12707.64   15341.33   9955.26   
2019-07              NaN    NaN  ...       7331.90   14796.05  24362.10   
2019-08              NaN    NaN  ...      11079.85   28956.08  16715.96   
2019-09              NaN    NaN  ...       2556.36     662.32       NaN   
2019-10              NaN    NaN  ...       2775.17        NaN       NaN   
2019-11              NaN    NaN  ...           NaN        NaN       NaN   
2019-12              NaN    NaN  ...           NaN        NaN       NaN   

                                                                           \
Cohort_Index              6         7         8         9        10        11   
Acquisition_Month                                                            
2019-01            51392.43  83164.89  49687.83  40313.12  45824.79  40757.46   
2019-02            41188.56  22313.65  44942.13  17146.07  28988.81  47910.20   
2019-03            70783.91  47920.00  39753.48  24759.69  36501.82       NaN   
2019-04            30148.90  15698.45  57088.59  18185.16       NaN       NaN   
2019-05            36035.79  37881.79  11455.74       NaN       NaN       NaN   
2019-06            36710.35  15475.08       NaN       NaN       NaN       NaN   
2019-07            29016.88       NaN       NaN       NaN       NaN       NaN   
2019-08                 NaN       NaN       NaN       NaN       NaN       NaN   
2019-09                 NaN       NaN       NaN       NaN       NaN       NaN   
2019-10                 NaN       NaN       NaN       NaN       NaN       NaN   
2019-11                 NaN       NaN       NaN       NaN       NaN       NaN   
2019-12                 NaN       NaN       NaN       NaN       NaN       NaN   


Cohort_Index             12  
Acquisition_Month            
2019-01            93254.57  
2019-02                 NaN  
2019-03                 NaN  
2019-04                 NaN  
2019-05                 NaN  
2019-06                 NaN  
2019-07                 NaN  
2019-08                 NaN  
2019-09                 NaN  
2019-10                 NaN  
2019-11                 NaN  
2019-12                 NaN  

[12 rows x 24 columns]
```

In [99]:
```
!pip install --upgrade seaborn
```

```
Requirement already satisfied: seaborn in c:\users\sinchan\anaconda3\lib\site-packages (0.12.2)
Collecting seaborn
  Using cached seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\sinchan\anaconda3\lib\site-packages (from seabor
n) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\sinchan\anaconda3\lib\site-packages (from seaborn) (2.1.4
)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\sinchan\anaconda3\lib\site-packages (from sea
born) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!=3.
6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib!=3
.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
!=3.6.1,>=3.4->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sinchan\anaconda3\lib\site-packages (from matplo
tlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas>=1.2->s
eaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas>=1.2-
>seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\sinchan\anaconda3\lib\site-packages (from python-dateutil>=2
.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Using cached seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.12.2
    Uninstalling seaborn-0.12.2:
      Successfully uninstalled seaborn-0.12.2
Successfully installed seaborn-0.13.2
```

In [114…

```
!pip install --upgrade seaborn matplotlib pandas
```

```
Requirement already satisfied: seaborn in c:\users\sinchan\anaconda3\lib\site-packages (0.13.2)
```

```
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\sinchan\anaconda3\Lib\site-packages)
WARNING: Ignoring invalid distribution ~atplotlib (C:\Users\sinchan\anaconda3\Lib\site-packages)
ERROR: Could not install packages due to an OSError: [WinError 5] Access is denied: 'C:\\Users\\sinchan\\anacond
a3\\Lib\\site-packages\\matplotlib\\ft2font.cp311-win_amd64.pyd'
Consider using the `--user` option or check the permissions.
```

```
Collecting matplotlib
  Using cached matplotlib-3.9.1-cp311-cp311-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: pandas in c:\users\sinchan\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\sinchan\anaconda3\lib\site-packages (from seabor
n) (1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib) (0
.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b) (4.25.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotli
b) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib)
(23.1)
Requirement already satisfied: pillow>=8 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib) (10.2
.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sinchan\anaconda3\lib\site-packages (from matplotlib
) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sinchan\anaconda3\lib\site-packages (from matplo
tlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas) (2023.
3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\sinchan\anaconda3\lib\site-packages (from pandas) (202
3.3)
Requirement already satisfied: six>=1.5 in c:\users\sinchan\anaconda3\lib\site-packages (from python-dateutil>=2
.7->matplotlib) (1.16.0)
Using cached matplotlib-3.9.1-cp311-cp311-win_amd64.whl (8.0 MB)
Installing collected packages: matplotlib
```

Insights

Customer Demographics and Behavior

In [ ]:
```
1 Gender Distribution: There are more female customers (63%) compared to male customers.
For all given cities, female customers are more than male customers.
2 Location Distribution: Most customers are from California (31%), followed by Chicago.
3 Customer Tenure: The average tenure is 25 months, with the most preferred tenure being 34
months. Tenure period ranges from 2 to 50 months.
```

Discount and Promotion Analysis

In [ ]:
```
1 Discount Distribution: Discounts range from 10% to 30%, with an average of 20%. Most
coupons were applied in January, with EXTRA10 being the most used coupon code.
2 Product Category Preference: The most purchased product category is Apparel.
```

Spending Analysis

In [ ]:
```
1 Spending Preferences: Customers prefer offline shopping over online shopping. Average
offline spend is higher than online spend.
2 GST Impact: GST ranges from 5% to 18%, with an average of 11.65%. Categories like
Drinkware, Fun, Lifestyle, and Apparel contribute more to GST.
```

Product and Sales Analysis

In [ ]:
```
1 Top-Selling Products and Categories: Top-selling products include Maze Pen, Google 22 oz
water bottle, Google sunglasses, etc. Top-selling categories by revenue include Nets-USA,
Apparel, and Office.
2 Delivery Charges: Most sales occur where delivery charges are low (0-10).
```

Hypothesis Testing Insights

In [ ]:
```
1 Discounts Impact on Revenue: Discounts have a significant impact on revenue.
2 Revenue by Gender: No significant difference in the average revenue generated by male and
female customers.
3 Revenue by Location: Significant difference in the average revenue generated by customers
from different locations.
4 Customer Tenure and Revenue: Significant correlation between customer tenure and revenue.
5 Marketing Spend and Revenue: Significant difference in the revenue generated by online and
offline marketing spend.
6 Delivery Charges and Revenue: Significant correlation between delivery charges and revenue.
```

Recommendation

Customer Demographics and Behavior

In [ ]:
```
1 Target Female Customers: Develop marketing campaigns specifically targeted towards female
customers.
 Tailor product offerings and promotions to female preferences.
2 Focus on Key Locations:
 Concentrate marketing efforts in California and Chicago.
 Implement location-specific promotions and localized advertising.
3 Enhance Customer Retention:
 Implement loyalty programs to reward long-term customers.
```

Discount and Promotion Strategies

In [ ]:
```
1 Strategic Discount Offering:
 Continue offering attractive discounts, especially at the start of
the year.

 Promote popular coupon codes and introduce similar high-demand
coupons.
2 Boost Apparel Sales:
 Increase inventory and variety in the Apparel category.

 Launch targeted marketing campaigns for Apparel products and
consider special promotions.
```

Spending and Revenue Optimization

In [ ]:
```
1 Strengthen Offline Marketing:
 Invest more in offline marketing channels and partnerships.

 Improve the online shopping experience to increase online spend.
2 Optimize Pricing Strategies:
 Consider the GST impact on high-GST categories.

 Educate customers on GST to justify pricing in these categories.
```

Product and Sales Enhancements

In [ ]:
```
1 Promote Top-Selling Products:
 Focus on promoting top-selling products and categories.

 Feature these products in promotions and marketing campaigns.
2 Reduce Delivery Charges:
 Consider subsidizing or reducing delivery charges to encourage
more purchases.

 Highlight low delivery costs in marketing materials.
```

Data Quality and Hypothesis Testing

In [ ]:
```
1 Regular Data Checks:
 Regularly check for missing data and ensure proper handling.

 Implement systems to minimize data gaps in future records.
2 Monitor Discount Effectiveness:
 Continue offering strategic discounts to boost sales.

 Monitor and adjust discount levels to maximize revenue.
3 Balanced Marketing Approach:
 Maintain a balanced marketing approach targeting both genders.

 Ensure product offerings appeal to a broad audience.
4 Location-Specific Strategies:
 Customize marketing strategies based on location-specific
insights.

 Focus on high-revenue regions and identify growth opportunities in
lower-revenue areas.
5 Increase Customer Tenure:
 Develop initiatives to increase customer tenure, such as loyalty
programs.

 Enhance long-term engagement strategies to boost revenue.
6 Optimize Marketing Spend:
 Allocate marketing budget effectively between online and offline
channels.

 Regularly analyze ROI (Return On Investment) to optimize spend
distribution.
7 Optimize Delivery Charges:
 Adjust delivery charges to enhance customer satisfaction and
increase sales.
```

In [ ]:
```
 Consider offering free or discounted delivery for larger orders.
```

In [ ]:

In [ ]: